



MID
the modeling company

Innovator**Business**

Innovator**Object**

Innovator**Data**

Innovator**Function**

Innovator**Report**

Innovator 11

Manual classiX - G Function

Innovator Function: Function-Oriented
Modeling with SA/RT/SD

Manual classiX - G Function-Innovator 11 (version 11.2)–Edition July 2010

Copyright © 1986-2010 MID GmbH Nuremberg, Germany. All rights reserved.

The copyright for Innovator software and the accompanying documentation is the property of MID GmbH.

Any reproduction or publication – in part or in whole – requires the written authorization of MID GmbH. Any violations of this will be prosecuted in accordance with copyright protection laws.

Registered trademarks and trade names are used in this documentation. Protection provisions apply to same, even if they are not marked as such.

The content of the documentation is for information purposes only and MID GmbH reserves the right to change it at any time without prior notice. MID GmbH will not be held liable for any damages arising from the use of the software or documentation.

Contents

Part G Innovator Function: Function-Oriented Modeling with SA/RT/SD.....	G-1
G-1 Modeling Systems Using SA Dataflow Diagrams	G-3
G-1.1 Creating and Renaming SA Diagrams.....	G-4
G-1.2 Working with Model Elements	G-5
G-1.2.1 Limiting the Use of Model Elements in Diagrams	G-5
G-1.2.2 Creating, Adding and Renaming Model Elements	G-7
G-1.2.3 Changing the Numbering of Processes	G-8
G-1.3 Displaying Data Transport in the Diagram.....	G-11
G-1.3.1 Inserting Flows	G-13
G-1.3.2 Changing Flows.....	G-15
G-1.3.3 Creating Split, Merge and Bi-Directional Flows.....	G-20
G-1.3.4 Changing Flow Direction	G-23
G-1.4 Defining and Specifying Data Structures in the Data Dictionary.....	G-25
G-1.4.1 Opening the Data Dictionary Editor	G-26
G-1.4.2 Loading Data Structures for Editing	G-27
G-1.4.3 Data Dictionary Syntax.....	G-28
G-1.4.4 Use and Definition of DD Entries	G-30
G-1.4.5 Finding Undefined Entries	G-32
G-1.4.6 Checking and Saving DD Entries	G-32
G-1.4.7 Renaming DD Entries	G-33
G-1.4.8 Deleting DD Entries	G-34
G-1.5 Decision Tables.....	G-34
G-1.5.1 Specifying Model Elements Using SA Decision Tables.....	G-34
G-1.5.2 Structure of the Decision Table	G-36
G-1.5.3 Formulating Conditions and Actions.....	G-39
G-1.5.4 Making Angle Brackets Visible	G-42
G-2 Using Real-Time Extension of Structured Analysis	G-43
G-2.1 Combining Flows in RT Decision Tables	G-43
G-2.1.1 Creating RT Decision Tables.....	G-44
G-2.1.2 Opening RT Decision Tables	G-44

G-2.2	Modeling System Behavior with State Diagrams	G-45
G-2.2.1	Creating State Diagrams	G-46
G-2.2.2	Specifying Possible States of the SA Diagram in the State Diagram	G-46
G-2.2.3	Inserting State Transitions	G-47
G-2.2.4	Changing Events or Actions.....	G-49
G-2.3	Creating Process Activation Tables for a Bar	G-50
G-2.3.1	Creating and Opening Tables.....	G-50
G-2.3.2	Showing and Hiding Column Groups	G-51
G-2.3.3	Defining Input and Output Signals for Process Activation	G-52
G-2.4	Creating Time Requirements Tables for a Terminator	G-55
G-3	Displaying Call Hierarchies Using Operation Diagrams.....	G-57
G-3.1	Creating and Renaming Operation Diagrams	G-58
G-3.2	Working with Operations and Data Areas.....	G-58
G-3.2.1	Creating, Adding and Changing Operations or Data Areas	G-59
G-3.2.2	Enabling Various Main Operations in the Model	G-61
G-3.2.3	Transforming External Operations or Data Areas into Internal Operations or Data Areas	G-62
G-3.2.4	Defining Formal Interfaces of Operations and Data Areas.....	G-63
G-3.2.5	Assigning SD Elements to SA Elements.....	G-66
G-3.3	Making Diagrams with Connectors Clearer	G-67
G-3.3.1	Creating and Renaming Connectors	G-69
G-3.3.2	Following Connectors.....	G-71
G-3.4	Displaying Calls of Operations and Data Areas	G-71
G-3.4.1	Creating an Operation's Calls	G-72
G-3.4.2	Setting the Order of Calls.....	G-73
G-3.5	Using Current Parameters	G-74
G-3.5.1	Creating Current Parameters.....	G-74
G-3.5.2	Changing Current Parameters.....	G-76
G-3.5.3	Changing the Order of Current Parameters.....	G-76
G-3.5.4	Assigning SD Modules to Elements	G-77
G-4	Merging Operations in Modules	G-79
G-4.1	Creating and Renaming Module Diagrams	G-79
G-4.2	Working with Modules.....	G-79
G-4.2.1	Creating, Adding and Changing Modules	G-80
G-4.2.2	Transforming External Modules into Internal Modules.....	G-81
G-4.3	Displaying Calls of Modules	G-82

G-5	Preparing Implementation.....	G-85
G-5.1	Making Settings for Implementation.....	G-86
G-5.1.1	Selecting the Implementation's Target Language.....	G-86
G-5.1.2	Influencing the Source Code Files' Names	G-87
G-5.1.3	Controlling Generation.....	G-88
G-5.1.4	Using Shortnames for DD Entries	G-91
G-5.2	Creating Implementations	G-92
G-5.2.1	Creating Include Files or Copy Books.....	G-92
G-5.2.2	To assign a DD entry to an include file or a copy book:	G-93
G-5.2.3	Creating Source Code Files and Transferring Model Data to the Implementation	G-96
G-5.2.4	Creating Other Source Code Files.....	G-98
G-5.2.5	Opening Implementations.....	G-99
G-5.3	Exporting Implementations in the File System.....	G-101
G-5.3.1	Automatically Exporting Implementations	G-101
G-5.3.2	Manually Exporting Implementations.....	G-102
Index		G-103

Part G Innovator Function: Function-Oriented Modeling with SA/RT/SD

Structured development is the optimum approach for both commercial (host-based) applications and technical applications (embedded systems).

The Innovator Function edition provides you with standard methods necessary for structured development (structured analysis (SA) and structured design (SD)) and offers you the security of a sound, tried and tested tool with real-time extension (RT) and a seamless connection with Entity Relationship Modeling (ERM).

Furthermore, Innovator Function integrates your existing structures and applications and facilitates their further development and maintenance. In addition to this, as the only CASE tool, Innovator Function offers a seamless connection to business process and data modeling.

You can use the Innovator Function edition to support the following phases of business process modeling or software creation:

- Analyze a system with the help of structured analysis (SA) as defined by De Marco and McMenamin/Palmer (see chapter G-1, "Modeling Systems Using SA Dataflow Diagrams", page G-3).
- Describe the system requirements.
- Describe the outside influences which affect the system or the influences which the system has on its surroundings. To do this, use the real-time extension (RT) defined by the Pirbhai/Hatley method (see chapter G-2, "Using Real-Time Extension of Structured Analysis", page G-43).
- Model the software architecture using structured design (SD) (see chapter G-3, "Displaying Call Hierarchies Using Operation Diagrams", page G-57).
- Support implementation by generating the system structure from structured design into the target language C or COBOL (see chapter G-5, "Preparing Implementation", page G-85).

G-1 Modeling Systems Using SA Dataflow Diagrams

Innovator supports the methods of structured analysis (SA) as defined by De Marco and McMenamin/Palmer. With the structured analysis even extensive, complicated facts can be illustrated simply and clearly. Its graphic elements mean that it is suitable for describing business processes or complex information and/or company information paths too.

Structured analysis displays the analyzed problem in a technologically-independent manner and without taking implementation aspects into account. To do this, use the following display formats:

- (data and control) flow diagrams
- Data dictionary
- Process and control specifications

For later software development, the structured analysis shows the requirements for the system to be implemented. Due to its independence from implementation, it is also suitable for communicating with people who do not work in the field of software development.

The result of structured analysis, the requirement characterization of the system, is formed using a hierarchical data flow diagram. The diagram at the very top is called a context diagram and displays the system's interface to the outside world.

It analyzes tasks in stages until there are only simple and easy to understand activities. This step from complex to simpler tasks is called refinement or decomposition and can be taken as often as required.

The SA diagram editor is a desktop which displays all model elements, as well as a hierarchy level's data and control flows; these can then also be edited here. In this chapter you will learn how to edit SA diagram's elements.

You can use the following elements to create a dataflow diagram in the SA diagram editor:

- Process (see page G-5)
- Terminator (see page G-5)
- Store (see page G-5)
- Bar (see page G-5)

Structured Analysis Application Field

Requirement Characterization Using Hierarchical Flow Diagrams

SA Diagram

SA Diagram Elements

- Data and control flows (see page G-11)

G-1.1 Creating and Renaming SA Diagrams

SA diagrams are hierarchical diagrams. There are two kinds of SA diagrams:

- **Context diagram:** This is the top-most diagram in the refinement structure. There is normally exactly one context diagram in a model. This, in turn, normally contains exactly one process.

You can normally only create the context diagram in the model browser; i.e. the top-most SA diagram of the system (see chapter B-2.2.2, "Creating and Opening Diagrams", page B-34).

- **Refinements diagrams:** The process/es in a context diagram is/are refined using a sub-diagram.

If you want to refine a process using an SA diagram, this is normally done in the SA diagram which contains the process (see chapter B-2.2.3, "Working with Hierarchical Diagrams", page B-35).

Other Information

- chapter B-2.3.1, "Changing Names, Relationships and Declarations", page B-49
- chapter B-2.5, "Removing and Deleting Elements", page B-61
- chapter B-4, "Working in Diagrams or Tables", page B-99

Diagram Types of the Refinement Diagram

Refinement diagrams can be further subdivided into three kinds:

- **Dataflow diagram:** Only the data flows are shown. All control flows, bars and control stores cannot be seen. No control flows or bars can be added in this mode.
- **Controlflow diagram:** Only the control flows are shown. All data flows and data stores cannot be seen. No data flows can be added in this mode.
- **Flow diagram:** Data flows and control flows are shown. No data flows, control flows or bars can be added in this mode.

How to proceed

To set a refinement diagram's diagram type:

- » Open the diagram (see page B-34) and select **Extras>Options....** The dialog box of the same name appears.
- » Using the junctions in the group box **Type**, set which diagram type you want to set and click on **[Apply]**.
- » Close the dialog box with **[OK]**.

G-1.2 Working with Model Elements

You can use the following model elements in the SA diagram:

A process is an activity which transforms your entered data into flow data. Either graphical decomposition into partial components (refinement diagram of the process), a decision table or a specification (text) describe how this transformation occurs.

Process

A terminator acts as an interface to the model's environment. It can be the starting or end point of both data and control flows. A terminator can normally only appear in the top-most diagram in the hierarchy—the context diagram. However, you can set that the terminator should also be able to be set in sub levels' diagrams (see chapter G-1.2.1, "Limiting the Use of Model Elements in Diagrams", page G-5).

Terminator

A store acts as a data repository which is processed in the system. Processes can be accessed for both reading and writing on a store.

Store

A bar (also known as a control process) serves as a relationship between a diagram and its control specifications. The control specification can take place using decomposition of a state diagram, a decision table and/or a process activation table. In addition to this, descriptive text can be stored. This serves as a source or target for control flows.

Bar

No more than one control specification exists for each SA diagram. Any number of bars can exist in an SA diagram to make things more clear. Each of these bars, however, represent the same control specification.

- chapter G-1.3, "Displaying Data Transport in the Diagram", page G-11.
- chapter B-4.3, "Working with a Diagram's Model Elements", page B-105
- chapter B-4.1, "Influencing how Diagrams or Tables are Displayed", page B-100

Other Information

G-1.2.1 Limiting the Use of Model Elements in Diagrams

The number of processes which you can insert in a context or flow diagram is limited as standard without needing to receive an appropriate message for a verification. The terminators can also only appear in the context diagram, not the flow diagram as standard. However, it is possible to modify these standard rules. These changes affect the entire model.

Note **Whether the highest permissible number is exceeded is not checked when creating model elements. A corresponding verification message only appears if you check the diagram (as long as you have not prevented this verification message).**

How to proceed

You require administrator rights to change the permissible number of processes and terminators.

- » Jump to the model browser and log-in as administrator (see chapter A-1.2.1, page A-5).
- » In the model browser, select [Model>Configuration>General Settings...](#)
The dialog box of the same name appears.
- » If applicable, select the tab [SA](#).

**Changing the
Maximum Amount of
Permissible Processes
in the Context
Diagram**

Only one process can exist in the context diagram as standard. This represents the entire model.

How to proceed

If you want to make various processes permissible in the context menu without having to first get an appropriate verification message upon verification:

- » Enter a number between 1 and 99 in the group box [SA constraints](#) in the field [CD processes](#).

**Changing the
Maximum Amount of
Permissible Processes
in the Flow Diagram**

Only nine processes can exist in a flow diagram as standard. If you want to include more than nine, check whether it is possible to combine two or more processes together in one refinement diagram.

How to proceed

If you also want to make various processes permissible in the flow diagram without having to first get an appropriate verification message upon verification:

- » Enter a number between 1 and 99 in the group box [SA constraints](#) in the field [Processes](#).

**Also Permitting
Terminators in Flow
Diagrams**

Terminators are only used in the context diagram as standard.

How to proceed

If you want to set that terminators should be able to be inserted in both context and flow diagram:

- » Activate the check box **Terminator** in the group box **SA constraints**.
- » Close the dialog box with **[OK]**.

G-1.2.2 Creating, Adding and Renaming Model Elements

You can either create model elements as new in the SA diagram or add existing model elements (see chapter B-2.1.2, "Logical and Graphic Existences of Model Elements", page B-22). You cannot copy model elements.

You can only create model elements in the model browser; they cannot be added.

Context diagrams can only be created in the model browser (see chapter G-1.1, "Creating and Renaming SA Diagrams", page G-4).

Note

If processes, terminators and stores are created once, they can be added to any number of diagrams and therefore display special aspects (see page B-22). Bars cannot be added.

**Adding Processes,
Terminators and Stores**

How to proceed

If you want to add a model element which already exists in the model:

- » Jump to the SA diagram and lock it.
- » Select **Insert><Element Type>>Add Existing...**
The dialog box of the same name appears.
All model elements which were already created in the model and do not exist in the current diagram are listed in the drop-down list **Name**.
- » Select the model element which you want to add to the current diagram from the drop-down list **Name**.
- » Click on **[Apply]** and close the dialog box with **[Close]**.

You can rename processes, terminators and stores in locked diagrams. For more information, read chapter B-2.3.1, "Changing Names, Relationships and Declarations", page B-49.

You can limit the names' drop-down list for stores.

**Renaming Processes,
Terminators and Stores**

How to proceed

To limit the drop-down list of names for stores:

- » Jump to the SA diagram and lock it.
- » Select the desired store and select `Edit>Properties....`
The dialog box of the same name appears.
The drop-down list `Name` contains all stores which do not already exist in the diagram.
- » If you want to limit the drop-down list to the names of the stores in the data dictionary, activate the option button `Data dictionary` in the group box `List entries`. You can limit the selection to `Data`, `Control` and/or `View` in the group box `Data dictionary type`.
- » If you want to limit the drop-down list to the names of the stores in the data dictionary, activate the option button `Other stores` in the group box `List entries`.
The respective limited drop-down list for names is provided for naming stores.

Selection in the group box `Effect` sets which elements are renamed in the model.

Other Information

- chapter B-2.2.4, "Creating Model Elements", page B-37
- chapter B-2.3.1, "Changing Names, Relationships and Declarations", page B-49
- chapter B-2.5, "Removing and Deleting Elements", page B-61

G-1.2.3 Changing the Numbering of Processes

You can influence the numbering of processes:

- You can change the way processes are numbered throughout the models.
- You can change the order of processes within a diagram.

Changing the Type of Numbering

Processes are numbered. All processes have an abbreviated form which has a number which is unique throughout all diagrams. This numbering does not contain information about whether (and where) the process is already being used elsewhere.

The unabbreviated syntax uses a decimal syntax for process refinements:

- The process in a context diagram has the number `0`.
- Processes in the top-most hierarchy level of the flow diagram are serially numbered with integers.
- Processes of the first refinement of a process contain the parent process' number and—separated by a full stop—a number which is unique throughout all diagrams: `[# Parent process].[# process]`.

- etc.

You can use this syntax to read how many refinement levels the process is currently contained in. The process' numbering which it was given when it was first created is also shown. This number is added to the current numbering in parentheses.

Settings made to the type of numbering has an effect throughout the models.

How to proceed

You require administrator rights to change the numbering of processes throughout the models.

- » Jump to the model browser and log-in as administrator (see chapter A-1.2.1, page A-5).
- » In the model browser, select **Model>Configuration>General Settings...**
The dialog box of the same name appears.
- » If applicable, select the tab **SA**.
You can set the type of process numbering in SA diagrams in the group box **Process numbering**:
 - **Short**: The processes of a hierarchy level are serially numbered in sequence.
 - **Long**: The processes contain a multi-digit number; the respective parent processes can be read off these. The number indicates a path component from the process of the highest hierarchy level to the process considered.
- » Select the type of numbering and close the dialog box with **[OK]**.

You can set which number each individual process should have within a diagram.

Changing the Order of Processes

How to proceed

To change the order within a diagram:

- » Lock the SA diagram.
- » Ensure that no diagram's element is selected.
- » Select **Edit>Properties...**
The dialog box of the same name appears.
- » Select the tab **Order**.
All processes of the current diagram are listed in the list box **Order**.
- » Mark one entry in the list and move it to the desired position using the **[Up]** or **[Down]** buttons.

- » Close the dialog box with [OK].
The new order is adopted. The processes' numbering is adapted in both the diagram and the model browser.

G-1.3 Displaying Data Transport in the Diagram

Flows connect model elements and illustrate data transport. Flows cannot be shown in the model browser.

Use data flows within SA diagrams. If you are working with real-time extension, you can also use control flows.

A straight line between two model elements indicates a simple data flow or control flow. It is made up of two parts—a starting and an end segment. A flow is given a name which defines its type. A flow can be assigned to a qualifier; this differentiates it from other flows of the same type. A solid line indicates a data flow and a dotted line indicates a control flow.

Data and Control Flows

Data flows can only be used in data flow diagrams and control flows can only be used in control flow diagrams. To be able to use both, the current diagram needs to be of the [flow diagram](#) type. Read how to change the diagram's type in section "Diagram Types of the Refinement Diagram", page G-4.

Note

A normal flow only has one name which applies to both segments. It can also be refined so that each segment has its own name and one segment is part of the other one.

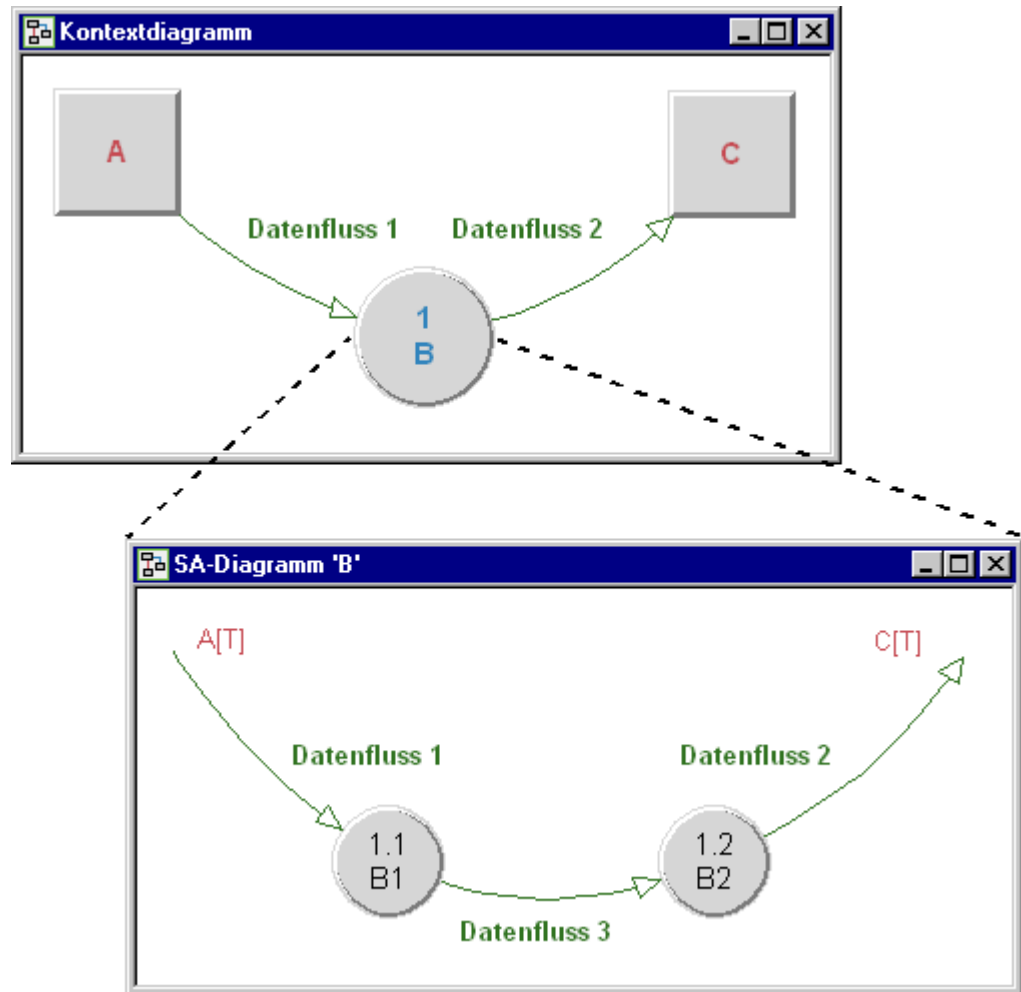
Refining a Flow

Flows from or to stores cannot be assigned names on the store side. They can be identified by the store which they are linked with. Refinements are permissible on the side leading out of the store; these then also need to be named.

Note

Environment flows are flows which flow from a higher level into the current diagram or which flow out of the current diagram into a higher level.

Environment Flow



Inconsistent Flow Inconsistent flows are flows which do not correspond to higher levels. They are marked on the environment side.

Displaying Flows Flows are displayed as directed arrows in diagrams. Flows which point in both directions are referred to as bi-directional flows. Flows are—as long as you have locked the diagram—split into two by a small circle. This enables so-called split and merge flows to be created: The split flow flows from a source element into one or more target elements, the merge flow flows from various source elements into one target element. Read more about split, merge and bi-directional flows in chapter G-1.3.3, page G-20.

Data Dictionary When creating a flow in a diagram, you also create a new entry in the data dictionary. You then need to specify this entry and explicitly create it (see chapter G-1.4, "Defining and Specifying Data Structures in the Data Dictionary", page G-25)

Other Information

- chapter B-4.3, "Working with a Diagram's Model Elements", page B-105

- chapter B-4.1, "Influencing how Diagrams or Tables are Displayed", page B-100
- chapter G-1.4, "Defining and Specifying Data Structures in the Data Dictionary", page G-25

G-1.3.1 Inserting Flows

You can create a flow either between two or more model elements or—as an environment flow—between a model element and its higher-level diagram.

By selecting the model elements involved, you set:

- Whether you create an environment flow or a data flow between model elements
- Whether this concerns a simple data flow or a split and/or merge flow.
- The direction in which the flow flows

You cannot connect the following model elements or diagrams using flows, as this is not permissible from a methodical view:

- Terminator and terminator
- Terminator and store
- Store and store
- Bar and bar
- Environment flow and store
- Environment flow and terminator
- Stores or terminators cannot exist on the source side of a merge flow.
- Stores or terminators cannot exist on the target side of a split flow.

Flows Which are Not Enabled

You can use the number of selected model elements to set whether you want to create an environment flow or a flow between model elements:

- Only select one model element (source) for an environment flow (from model element in the environment).
- If necessary, reverse the direction of the environment flow at a later stage (see section "Changing the Direction", page G-23).

The direction of the flow is set by the order in which the two model elements are selected.

- Select precisely two model elements for a simple data flow or control flow: First the source element and then the target element.
- For a split flow, first select the source element and then the target element.
- For a merge flow, first select any target element and then all other target elements and the source element. Reverse the direction of the flow at a later stage (see section "Changing the Direction", page G-23).

Inserting Flows

Note **If there is a bar below the selected model elements, you can only create a control flow; a data flow cannot be created.**

You can use splits and merges to create other combinations (see section "Changing Flow Direction", page G-23).

How to proceed

To insert a flow (between model elements) or an environment flow:

- » Lock the diagram.
- » Select the model element which should be the source of the flow.
- » Select [Insert>Data Flow>From Environment](#) or [Insert>Data Flow>To Environment](#) an/or [Insert>Control Flow>From Environment](#) or [Insert>Control Flow>To Environment](#) to create an environment flow.
- » If you do not wish to create an environment flow, also select the model element/s which should be the target of the flow and select [Insert>Data Flow>Internal](#) or [Insert>Control Flow>Internal](#).
The data flow and/or control flow is inserted. It has a default name.

Showing Element Names in the Environment Flow

Environment flows show that an element in the parent diagram is refined by the current diagram. The flow which flows into the element there is indicated as an environment flow in the refinement diagram. You can set whether the element's name from the parent diagram should be shown in the environment flow for the entire model or not.

How to proceed

To show the element names from the parent diagram in the refinement diagram's environment flow:

- » Jump to the model browser and log-in as administrator (see chapter A-1.2.1, page A-5).
- » Select [Model>Configuration>General Settings...](#)
The dialog box of the same name appears.
- » If applicable, select the tab [SA](#).
- » Activate the check box [Show origin](#) in the group box [Environmental flows](#) and close the dialog box with [\[OK\]](#).
The name and type of the corresponding element is now displayed in the environment flows in the refinement diagrams. The type appears in square brackets after the name (e.g. [T] for terminator).

Other Information

- chapter G-1.3.3, "Creating Split, Merge and Bi-Directional Flows", page G-20

- chapter G-1.3.4, "Changing Flow Direction", page G-23
- chapter B-2.5, "Removing and Deleting Elements", page B-61

G-1.3.2 Changing Flows

You can change the flow's properties at a later stage. The following changes are possible:

- Renaming flows (see page G-15)
- If applicable, assign a qualifier (see page G-16)
- Setting decompositions in the data dictionary (see page G-17)
- Changing flow direction (see page G-23)
- Changing the type of flow (control or data flow), as long as this takes place in a flow diagram (see page G-18)

You can also change the type for various flows at the same time. Only one flow can be selected for all other properties.

How to proceed

To change a flow:

- » If necessary, lock the diagram.
The flow's selection points are visible.
- » Select a flow and select [Edit>Properties....](#)
The dialog box of the same name appears.

You can rename flows. Similar to renaming model elements (see chapter B-2.3.1, "Changing Names, Relationships and Declarations", page B-49), you need to take into consideration the effects on the other flows that exist.

There are various different ways to select a new name:

- A: Look through the complete list of all flows.
- B: Limit the list to data flows which Innovator recommends (see page G-16).
- C: Show the DD entries above or below the current flow (see page G-17).
- D: Enter a new name.

Renaming Flows

Flows from or to stores cannot be assigned names on the store side as they are identified by the store which they are linked with.

Note

Limiting Drop-Down Lists

Depending whether you have selected a flow between two model elements or an environment flow in the diagram, various alternatives are offered in the dialog box **Edit/Properties** in the tab **Name** in the group box **List**. You can limit the names' drop-down list using these alternatives.

- **Limiting names of flows between model elements:**

- **All names in the model**

- All the model's flows and data dictionary entries are shown.

- **Names of corresponding environment flows in source processes**

- All the flows which flow out of the source element's refinement diagram are shown.

- **Names of corresponding environment flows in target processes**

- All the flows which flow into the source element's refinement diagram are shown.

- **Limiting the name of the environment flow:**

- **All names in the model**

- All the model's flows and data dictionary entries are shown. If this concerns a data flow (not a control flow), you can limit the data dictionary entries to entries of the **View** type.

- **Names of corresponding flows in diagram**

- Depending on the direction of the environment flow, the flows which flow in and out of the parent process in the parent diagram are shown.

- **Names of corresponding environment flows in source processes**

- All environment flows from the refinement diagrams to model elements concerned are shown as long as they have the correct direction (dependent on the selected flow direction).

How to proceed

If you want to select a name:

- » Select the tab **Name** in the dialog box **Edit/Properties**.

- The default name of the selected flow appears in the field **Name**.

- The drop-down list contains all flows and all data dictionary entries.

- » Firstly, limit the drop-down list by activating an alternative in the group box **List**—the check box **Add views to list** for the option button **All names in the model** if necessary.

- The names list is adapted according to the selected constraints.

- » Select a name from the drop-down list **Name** or enter a new name.

- As well as assigning the name, you can also assign a data flow to a qualifier to limit the flow's value range. You can enter the qualifier's name or select it from a list. This list contains the flow's qualifier which you limited as described above. Entering a qualifier is optional.

- » Jump to the tab **Qualifier**.

- » Select a name for a qualifier from the drop-down list `Qualifier` if necessary or enter a new qualifier.

You can use radio buttons and check boxes in the group box `Effect` to set where the name change should have an effect:

- `Name of the selected flow`
Only the name of the current flow is changed.
- `Names of corresponding flows in diagram`
The name of all flows with the same name is changed in the current diagram.
- `Names of corresponding flows in model`
The name of all flows with the same name is changed in all of the model's diagrams.
- `Names of corresponding flows in source process refinements`
All flows which flow out of the source element's refinement diagram are renamed.
- `Names of corresponding flows in target process refinements`
All flows which flow into the source element's refinement diagram are renamed.
- `Maintenance in specifications, tables and state diagram`
The references in specifications and in the RT area are renamed.

Setting Effects

How to proceed

To set to what extent renaming the flow should have an effect:

- » Activate the desired radio button or check box in the group box `Effect`.
- » If you do not want to rename references in the specifications, deactivate the check box `Maintenance in specifications, tables and state diagram`.
- » Click on `[Apply]`.

If you need a data dictionary entry as a name, you can display the flow's substructures and higher structures. Read how to work with the data dictionary in chapter G-1.4, "Defining and Specifying Data Structures in the Data Dictionary", page G-25.

Using Data Dictionary Entries

How to proceed

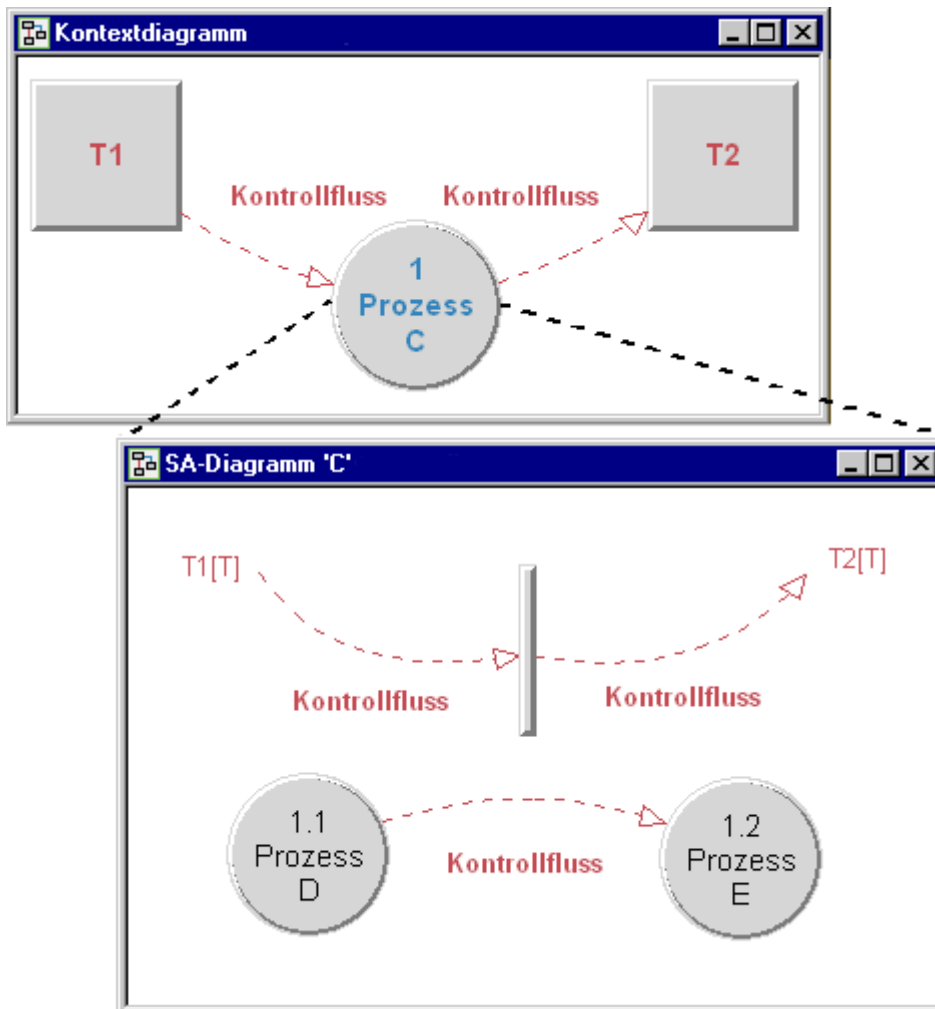
To use data dictionary entries:

- » Select the tab **Decomposition** in the dialog box **Edit/Properties**. You can use the **[DD references]** and **[DD refinements]** buttons to control which DD entries you want to display:
 - **[DD references]**
Entries which contain the current DD entry as a refinement
 - **[DD refinements]**
All refinements of the current DD entry
- » Click on one of these buttons.
The corresponding entries are shown in the list.
- » Click on the entry to select an entry from the list as a name.
A selection in the DD list is adopted in the field **Name**. According to the button which was last pressed, the newly selected flow's DD entries are displayed in the DD list.
- » If you want to assign a name according to the data dictionary's entries, click on **[Apply]**.

Changing Flow Types

You can use the flow type to specify whether the flow is a data flow or a control flow. A flow which touches a bar is always a control flow. The situation where a control flow exists between two processes can also occur.

Note **Data flows can only be used in data flow diagrams and control flows can only be used in control flow diagrams. To be able to use both, the current diagram needs to be of the **flow diagram** type. Read how to change the diagram's type in section "Diagram Types of the Refinement Diagram", page G-4.**



How to proceed

To change the flow type:

- » Select the tab **Properties** in the dialog box **Edit/Properties**.
- » Activate the option button **Data** in the group box **Flow type** for a data flow or activate the option button **Control** for a control flow.
- » Set to what extent the change should have an effect using the radio buttons in the group box **Effect** (see page G-17).
- » Click on **[Apply]**.
- » Once you have made all changes, close the dialog box with **[OK]**.

G-1.3.3 Creating Split, Merge and Bi-Directional Flows

Data and control flows can links more than two model elements.

Split Flows A flow which only has one start segment, but various end segments specifies a split flow. It is irrelevant whether these end segments are refinements of the flow or the complete flow itself flows in other directions.

Merge Flows A flow which has various start segments, but only one end segment specifies a merge flow. It is irrelevant whether these start segments are refinements of the flow or the complete flow itself flows in from various directions.

You can change the flow's structure at a later stage. The following changes are possible:

- Creating split flows (see section "Creating Split Flows", page G-20)
- Creating merge flows (see section "Creating Merge Flows", page G-21)
- Splitting a flow made up of various parts into sub flows (see section "Splitting Merge, Split or Bi-Directional Flows into Sub Flows", page G-23)

How to proceed

The current diagram must be locked during all of these changes.

- » Lock the diagram.
The flow's selection points are visible.

Creating Split Flows The following options are available to you if you want to create a split flow:

A: Create a new end segment for a simple data flow which already exists.

B: Merge the start segment of two data flows into one start segment (see page G-21).

Option A: The following constraints need to be taken into consideration if you want to create a split flow from a simple flow by adding a new segment to the target element.

- None of the model elements marked can already be linked with the flow's source.
- A flow's target cannot be a store or a terminator.
- It cannot contain any impermissible relationships (see section "Flows Which are Not Enabled", page G-13).

How to proceed

To create a split flow by adding a new end segment:

- » Select the flow's start segment.
- » Also select the model element which the new segment should lead to.
- » Select **Edit>Split**.

The flow is forked at the selection point. The new end segment is linked with the selected model element.

The following constraints need to be taken into consideration if you want to create a split flow from two flows by merging the inflowing segments:

- The start segments of both flows have the same source.
- The start segments must have the same name.
- Neither of the start segments is part of a merge flow.

Option B:

How to proceed

To create a split flow from two single flows

- » Select the flow's start segments which you want to merge.
- » Select **Edit>Merge**.

The marked segments of both flows are consolidated into one segment.

The following options are available to you if you want to create a merge flow:

Creating Merge Flows

A: Create a new start segment for a simple data flow which already exists.

B: Merge the end segment of two data flows into one end segment (see page G-22).

The following constraints need to be taken into consideration if you want to create a merge flow from a simple flow by adding a new inflowing segment.

Option A:

- None of the model elements marked can already be linked with the flow's target.
- A flow's source element cannot be a store or a terminator.
- It cannot contain any impermissible relationships (see section "Flows Which are Not Enabled", page G-13).

How to proceed

To create a merge flow by adding a new start segment:

- » Select the flow's end segment.
- » Also select the model element which the new segment should come from.
- » Select [Edit>Merge](#).

The new segment leads from the selected model element to the flow's selection point.

Option B: The following constraints need to be taken into consideration if you want to create a merge flow from two flows by merging the out flowing segments:

- The end segments of both flows have the same target.
- The end segments must have the same name.
- No flow is a split flow.

How to proceed

To create a merge flow from two single flows

- » Select the flow's end segments which you want to merge.
- » Select [Edit>Merge](#).

The end segments are merged.

Creating Bi-Directional Flows

To create a bi-directional flow from two single flows.

The following constraints need to be taken into consideration:

- Two flows run between the same model elements.
- Both flows run in opposite directions.
- Both flows have the same name.

Read how to create a bi-directional flow from a uni-directional flow on page G-23.

How to proceed

To merge two flows into one bi-directional flow:

- » Select both flows using their selection points.
- » Select [Edit>Merge](#).

The two flows which are flowing in opposite directions are merged into one bi-directional flow.

You can split merge, split and bi-directional flows and, in doing so, create various individual uni-directional flows

Splitting Merge, Split or Bi-Directional Flows into Sub Flows

How to proceed

To split a flow made up of various parts into single flows:

- » Select
 - The end segment for a merge flow
 - The start segment for a split flow
 - The selection point for a bi-directional flow
- » Select **Edit>Split**.
The flow is split into the sub flows concerned. A bi-directional flow is replaced by two uni-directional flows.

G-1.3.4 Changing Flow Direction

You can change the flow's direction structure at a later stage. The following changes are possible:

- Reversing the direction of the flow
(see section "Changing the Direction", page G-23)
- Linking the flow segment with other model elements
(see section "Reconnecting Flow Segments", page G-24)

How to proceed

The current diagram must be locked during all of these changes.

- » Lock the diagram.
The flow's selection points are visible.

The direction of the flow is set by the order in which the two model elements are selected. You can still change this direction. Only either data flows or control flows can be reversed. The arrow can flow in both directions.

Changing the Direction

All segments from each reversed flow must be marked.

Note

How to proceed

To change the flow's direction:

- » Click on the selection point of one or more flows of the same type.
All of the flow's segments are selected.
- » Select **Edit>Properties...**
The dialog box of the same name appears.
- » Select the tab **Direction**.
An alternative is specified for each direction in the group box **Direction**:
 - option button **von <bish. Quellelement>**
 - option button **zu <bish. Quellelement>**
- » If you want to exchange source and target elements, activate the option button **zu <bish. Quellelement>**.
- » If you want the flow to flow in both directions, activate the option button **Bidirectional**.
- » Close the dialog box with **[OK]**.
The dialog box closes, all selected flows are reversed.

Reconnecting Flow Segments

You can connect a flow's segment with a model element at a later date. You can also redirect a flow from a model element to an environment. Make the new connection individually for each flow segment.

The following constraints need to be taken into consideration:

- None of the flow's segments are already connected to the target.
- It cannot contain any impermissible relationships (see section "Flows Which are Not Enabled", page G-13).
- Segments from split and merge flows cannot be connected to the environment, stores or terminators from the branched side.

How to proceed

To connect a flow segment with another model element:

- » Select the segment which you want to connect with another model element.
- » Also select the element if you want to link the segment with a model element.
Do not enter a model element if you want to create an environment flow.
- » Select **Edit>Reconnect**.
If a model element was selected, the flow segment is connected with this; if this is not the case, then the flow segment is connected with the environment.

G-1.4 Defining and Specifying Data Structures in the Data Dictionary

The data dictionary (DD) is an important Innovator Function edition tool.

The data dictionary defines data flows, control flows and stores. An entry must be made in the data dictionary for each data and control flow diagram in the model. Flow definition occurs by means of a syntax adjusted to the Backus-Naur form.

An analysis diagram does not identify what structure the data which is transported in the flow of the diagram has. It is clear that a flow or store (e.g. with the name `list`) transports or stores data from the type `list`. The exact description of such a type can be found in the data dictionary. It is, therefore, insignificant whether the defined type is already used in the analysis diagram or not. A consistency test registers whether a type is defined for every flow and store in the diagram or whether every defined type is used in the diagram.

Various editors are available within the Innovator Function edition for editing the data dictionary. You can modify existing type definitions or create new ones there and transfer them to the data dictionary upon successful syntax checking.

You need to keep to a simple syntax when defining data types. Adherence is observed by the system (see chapter G-1.4.3, "Data Dictionary Syntax", page G-28).

A data dictionary entry is automatically created upon creation of a flow or store.

You can make the following modifications in the data dictionary editor (DD Editor):

- Access information about an entry.
- Create a new entry.
- Load and edit an entry.
- Rename an entry.
- Delete an entry from the data dictionary.

You can use various functions for checking the data dictionary:

- Search for an entry which has not been defined yet.
- Check the syntax of the definition.

When using Innovator CASE tools for developing a software system's development, the tools for structured analysis (SA) and (structured) entity relationship modeling (ERM/SERM) can work together. You can produce a relationship between data specifications in the SA and views from the conceptual schema in the ERM/SERM.

A data structure specification is generated in the DD syntax from a complete view group in the ERM/SERM and stored in the data dictionary. A `∇` (view) indicates these structure specifications. These types of DD entries

Relation with ERM/SERM

cannot be changed to ensure consistency in the data model. Other than this, these data dictionary entries can be used perfectly normally to e.g. split data flows and send these parts to various processes.

Note **As this link requires a running server for the corresponding data model (which cannot be demanded for every other editing), any changes made to the views in the SERM/ERM section are not automatically incorporated into the SA. The update takes place by calling a Tcl script in the SA/SD or ERM model tree.**

An additional text called **COBOL88** can be configured for supporting COBOL 88 levels whose contents is used for the generation. The contents of the text can be automatically updated during the view connection described above. An additional text called **COBOL88** which is added to the data element is also used in the SERM/ERM.

G-1.4.1 Opening the Data Dictionary Editor

The data dictionary is normally opened via a diagram. Depending on which element you have selected, you will obtain the following entries:

- Name of the flow segment
- Name of the flow segment, which is connected with certain model elements
- Name of all flow segments available in the diagram

You can select whether you only wish to open the DD entry for the selected element, or if you want to be able to see the descriptions of its components as well.

Note **You can open the data dictionary repeatedly in succession. The new entry is added in the same window of the entry which is already open.**

How to proceed

To edit the data dictionary (the diagram does not have to be locked to do this):

- » Select the flow or model element which you want to see the definition of. If you want to get see definitions for all elements in the diagram, make sure nothing is selected in the diagram.

You can either open only the definition of the current element or the definition of the current element and its components as well.

- » If you only want to open the definition of the current element in the diagram, select **Jump>To Data Dictionary>Pure**.
A data dictionary window is opened. It contains DD entries, which you have specified through the selection.
- » If you want to open a component at the same time as an element definition, select **Jump>To Data Dictionary>Complete**.
A data dictionary window is opened. It contains DD entries, which you have specified through the selection. The components which belong to it are also opened.

Data dictionary entries are displayed and modified in a text window, the data dictionary editor. The text window takes up a large part of the window. You can load and modify type definitions here.

Data dictionary Editor

G-1.4.2 Loading Data Structures for Editing

You can display other entries once the data dictionary is opened. There are two options available for this:

A: Jump to the diagram and open the data dictionary for another element (see chapter G-1.4.1, "Opening the Data Dictionary Editor", page G-26).

B: Load the entry within the editor window.

If you select option A, the new entry is always added to the end of all entries which are already displayed. If, on the other hand, you choose option B, the new entry is inserted at the insertion point.

Note

You can empty the editor window before loading an entry. If you have made the changes in the current entries, a corresponding message appears.

Emptying the Editor Window

If you save changes, the editor window is automatically emptied.

Note

How to proceed

To empty the editor window:

- » Select **Content>Clear Window** in the DD editor.
If no unsaved entries were displayed, the editor window is emptied. Otherwise, a corresponding message appears.

- » Click on **Yes** in the message box to save the changes. Click on **No** if you want to reject the changes.
The contents of the text window then has the status **unaltered** once deletion has been completed.

Inserting Entries at the Insertion Point

You can specify where the new entry should appear in the editor window.

How to proceed

If you want to load another entry from the DD editor:

- » Position the insertion point in the DD editor where you want to insert the entry.
- » Select **Content>Load Definition...** in the DD editor.
The dialog box of the same name appears. All DD entries which are already defined are listed in the drop-down list **Name**
- » Activate the check box **Complete** if you want the new entry to also contain the description of its components.
- » Limit the list of names if necessary using the check box in the **group box What**.
You can limit the list by entering search criteria using wildcards (see chapter B-3.3, "Searching for Strings using Text Samples", page B-95).

Note **If a pattern is entered in the field **Name**, all suitable entries are loaded.**

- » Enter the name for the new DD entry in the **field Name** or select one from the drop-down list.
- » Close the dialog box with **[OK]**.
The entries should be locked if this is possible and are then loaded in the text window:
 - If the **Complete** mode is deactivated, only the selected DD entries are loaded into the text window.
 - If the **Complete** mode is activated, both the selected DD entries and all entries which are needed for the definition are loaded.All entries which the user does not have access to for changing are marked as comments with **!***** in front.

G-1.4.3 Data Dictionary Syntax

The data dictionary's syntax permits the composition of data and control types of other types in different ways.

Data types cannot be composed of control types and control types cannot be composed of data types. The first letter identifies the type of entry:

- D: Data flow
- C: Control flow
- V: View

As well as the composed types, terminal types, so-called literals, can also be defined. Each type defined in the data dictionary (also spanning various levels) should ultimately be composed of literals.

An entry in the data dictionary is called a type definition and conforms with the structure of the following syntax diagrams.

Name	Icon	Description/version
Definition	=	Described by, comprises
Sequence	+	Together with A sequence permits the composition of one type made up of multiple components into several types.
Junction	[... ...]	Either... or.. Square brackets enclosing an enumeration of types indicate a junction. Every object of a defined type contains a component of any selected type of the enumeration.
Repeat	m{ ... }n	A repeat of at least m-times to no higher than n-times The designation of m and n is optional. A type name in curly brackets means that the components of this type will be used more than once in the configuration of the defined type (<i>iteration</i>). The number of iterations can be set using indices. A number before the parenthesis is interpreted as the minimum number of the iterations. A type name before the parenthesis is often interpreted as a DEPENDING-ON variable for the COBOL language. A number after the parenthesis is interpreted as the maximum number of the iterations. A type name is interpreted as define for the language C in particular; its contents is used as the maximum number.
Option	(...)	Is available A type name in parentheses within a type definition means that a component of this type can be (but does not have to be) involved in the composition of the defined type.
Terminal Components (Literal)	" ... "	No longer a defining term
Comment	* ... *	Own description can be written, note

Name	Icon	Description/version
End of Definitions	.	Conclusion of definition
Software Name	* <name> *	The software name can be appointed in the comment You can input a short name in angle brackets within the comment which will be used in place of the type name for generating the implementation.

A (name) tag may not contain more than 99 characters. Longer tags are abbreviated to this length.

This name must be contained within ' . . . ' so that data flows can be described in the data dictionary's syntax using special characters or spaces. Umlauts also count as special characters.

G-1.4.4 Use and Definition of DD Entries

You can use the DD editor to display information about all DD entries which already exist:

- Information about the modification time of the entries which are currently displayed.
- Detailed Display of How the DD Entries are Structured
- Displaying additional information about DD entries, such as model elements concerned and modification history.

Identifying the Modification Status

There are two ways to set whether changes were already made to the current entries:

- **Modified** can be seen in the status bar if changes have already been made, or **Not Modified** if the display corresponds to the saved entry.
- The **Data Dictionary from...** data dictionary entry can be seen in the screen window's title row. If this entry appears in square brackets this means that the contents of the respective text window is in the **Modified** state, i.e. modifications have not yet been saved.

Opening Information about the Structure of a DD Entry

The simple display of information about a DD entry comprises the entry's structure and, if applicable, its components. A structure which is similar to a tree is displayed for each component and also for its construction—right down until the Literals.

How to proceed

To call the definition information:

- » Jump to the data dictionary editor and select **Extras>Information...**

The dialog box of the same name appears. It corresponds to the dialog box **Content/Load Definition** (see section "Inserting Entries at the Insertion Point", page G-28).

- » Enter a name or select one from the list.
- » Close the dialog box with **[OK]**.

Depending on the **Complete** mode, the entries are displayed in the information window:

- If the mode is deactivated, only the selected DD entries are displayed.
- If the mode is activated, both the selected DD entries and all entries which are needed for the definition are displayed.

If you want to get information about model elements which are linked with an entry, call the detailed information. You can also get modification history and bookmarks for the entry here.

Opening Information About Model Elements Involved

How to proceed

To get detailed information about the entry:

- » Jump to the data dictionary editor and select **Extras>Information...**

The dialog box of the same name appears. It corresponds to the dialog box **Content/Load Definition** (see section "Inserting Entries at the Insertion Point", page G-28).

- » Enter a name or select one from the list.
- » Close the dialog box with **[OK]**.

The following data is shown in the information window for each selected entry:

- Entry's name
- Modification history
- List of all model elements which have a flow of this type which flows into it
- List of all model elements which have a flow of this type which flows out of it
- List of all the design's model elements which use a current parameter with the flow's type
- List of all bookmarks with values which are defined for the entry

G-1.4.5 Finding Undefined Entries

All DD entries which do not yet have a description can be automatically listed. The editor window must be empty for this.

How to proceed

To show the undefined entries:

- » Empty the editor window (see section "Emptying the Editor Window", page G-27)

- » Select [Engineering>Edit Undefined Flows](#).

A DD entry is generated in the text window for each of the model's undefined data flows or control flows and is attached to the current contents of the text window.

The entry identifies the flow's type as data type or control type accordingly.

The entry which is generated is syntactically correct but empty.

Note As the entries are syntactically correct you can save these without an error message appearing. In doing so however, you create entries. This means that the entries can no longer be found using the [Engineering>Edit Undefined Flows](#) menu item.

- » Make the necessary changes to each undefined entry and save them (see page G-32).

G-1.4.6 Checking and Saving DD Entries

If you have made any modifications to an entry then you need to save it. A syntax check is also automatically conducted.

How to proceed

To save your modifications:

- » Make sure that you have the necessary authorization for the data dictionary.

- » Select [Content>Write to Dictionary](#).

The contents of the text window will be checked for syntax errors and some semantic errors and those which are caused by the system (e.g. access clashes).

Syntax error messages appear directly below the screen row concerned. Semantic and system error messages appear at the end of the text.

Error messages are highlighted with !*. Do not delete these before the parser has been rerun. Apart from this, they messages can be disregarded.**

Note

If the parser runs without any errors, then the text window contents will be transferred into the data dictionary's database and the window emptied.

How to proceed

To correct an error-free entry,

- » Make the necessary corrections and save them.
The parser is then rerun. If your changes were accurate, then no error messages will appear.
- » Carry out these steps until no more error messages appear.
The editor window is emptied and the contents of the text window is transferred into the data dictionary's database.

G-1.4.7 Renaming DD Entries

You can rename a DD entry. This sets whether only the DD entry should be changed or whether flows with the same name or existences in tables and texts should also be changed.

How to proceed

To rename a DD entry:

- » Jump to the data dictionary editor and select [Content>Rename Entry...](#)
The dialog box of the same name appears.
- » Limit the list of names if necessary using the check boxes in the group box [What](#).
You can limit the list by entering search criteria using wildcards (see chapter B-3.3, "Searching for Strings using Text Samples", page B-95).
- » Enter the name for the new DD entry which you want to change in the field [Name](#) or select one from the drop-down list.
- » Enter the new name of the entry in the field [New](#).
- » Activate the check box [References](#) if you want to adopt the change to all the entry's existences in diagrams, tables and texts.
- » Close the dialog box with [\[OK\]](#).

G-1.4.8 Deleting DD Entries

If you want to delete a DD entry you need to decide whether only the entries which aren't used anywhere should be affected or also those which are being used.

How to proceed

To delete a DD entry:

- » Jump to the data dictionary editor and select **Content>Delete From Dictionary...**
The dialog box of the same name appears.
- » Enter the name for the new DD entry which you want to delete in the field **Name** or select one from the drop-down list.
- » To only delete those entries which aren't used anywhere, activate the check box **No usage**.
If the deletion control is turned on, the dialog box **Edit/Delete** appears. All objects which are listed are deleted by the action.
- » Check whether you really want to delete the entry. If necessary, cancel the procedure with **[Cancel]**.
- » Otherwise, close the dialog box with **[OK]**.
The DD entry is deleted.

G-1.5 Decision Tables

You can use decision tables to control which actions are carried out if certain conditions are met.

Decision tables are used in structured analysis to specify processes. You can also specify an SA diagram's control processes (bars) in the real-time extension of structured analysis.

Read how to create SA decision tables for processes in chapter G-1.5.1, page G-34.

Read how to create RAT decision tables in chapter G-2.1.1, page G-44.

G-1.5.1 Specifying Model Elements Using SA Decision Tables

A text editor is provided for specifying a model element (see chapter C-2, "Specifying Elements", page C-9). If you want to specify a process, you can also use an SA decision table instead of the text.

Read how to use tables for specifying bars in chapter G-2.1, "Combining Flows in RT Decision Tables", page G-43. Read how to use tables for specifying terminators in chapter G-2.4, "Creating Time Requirements Tables for a Terminator", page G-55.

SA decision tables provide process specification. They supplement or replace text description of processes. Assignment to the SA process takes place using name similarity. Read how to work with decision tables in chapter G-1.5, "Decision Tables", page G-34.

You can create an SA decision table for a process in both the model browser and in the SA diagram:

Creating SA Decision Tables

How to proceed

To create an SA decision table in the diagram:

- » Select the process in the diagram. Ensure that only this model element is selected.
- » Select **Engineering>Create Decision Table**.
The table is created and opened.

How to proceed

To create an SA decision table in the model browser:

- » Select the package in the model browser in the model tree which contains the process.
- » Select the process in the list of model elements.
- » Select **Element>New>SA Object>SA Decision Table**.
The table is created and opened.

G-1.5.2 Structure of the Decision Table

The general structure of a decision table is as following:

		1	2	...	n	Sonst
Condition part	Bedingung 1	-	-	-	-	
	Bedingung 2	-	-	-	-	
	...	-	-	-	-	
Action part	Aktion 1					
	Aktion 2					
	...					

Note A newly-created decision table only contains one row for the condition part and the action part respectively. Once you have inserted other lines, division becomes apparent.

Condition Part The first column contains the conditions in the condition part (e.g. `<age> < 6 years>`).

The numbered columns represent the rules. A rule cell in the condition part contains in each case one possible case of the assigned condition i.e. the case in that line (e.g. `Yes`). One rule makes up a combination of all conditions in the condition table. It is completed once all conditions accept states which correspond to the rule assigned conditions.

The table should be constructed in such a way that precisely one rule is implemented for every possible combination of states. The `else` column is optional. It covers all combinations which are not implemented by a rule. The `else` column will, therefore, only have empty entries in the condition table.

Possible actions are listed in the first column in the action part (e.g. `<o Admission charge> = 0`). If one of the implemented rules' cells is marked with a cross in the action part, then the action in that line will be activated. If there is more than one cross in a column, then more than one action will be carried out.

Action Part

In its simplest form, a condition is a statement which is either true or false. There are then three possible states:

Formulating Conditions

- **Y** (the statement is true)
- **N** (the statement is false)
- **-** (it is irrelevant whether the statement is true or false)

The cell values are fixed for this table type in Innovator interfaces.

Note

A simple air conditioning unit should regulate the temperature throughout the day to a value between 18 and 22 degrees Celsius. The temperature at night should be less than 18 degrees Celsius.

Example

	1	2	3	Else
<i Time of day> is nights	N	N	Y	
<i Temperature> < 18	Y	N	N	
<i Temperature> > 22	N	Y	-	
<o increase heat output>	■			
<o maintain heat output>				■
<o reduce heat output>		■	■	

There is a second type of decision table which does not necessarily have a statement in the first column in the condition part, but will have any text, e.g. the name of the system's input size instead. Together with the cell value which can adopt any text here, this forms the condition.

The cell values are not fixed for this table type in Innovator interfaces.

Note

Example The example above changed using this table type as follows:

	1	2	3	Else
<i Time of day>	Day-time	Day-time	Night-time	
<i Temperature>	< 18	> 22	> 18	
<o Increase heat output>	■			
<o Maintain heat output>				■
<o Reduce heat output>		■	■	

Example The admission charge for a circus should be set. This value depends on the entry parameters *age*, *profession* and *time of day*. The following conditions apply for pricing:

- The basic charge is 10 euros.
- Children under the age of 6 get in free.
- Children up to the age of 14 pay 6 euros.
- Children, students etc. get 2 euros off the admission charge in the afternoons. This price reduction does not apply for the evening performance.

The following SA decision table with the cell values with the *fixed* type correctly reproduces the data described:

	1	2	3	4	Else
<i Age> < 6 years	Y	N	N	N	
<i Age> < 14 years	-	Y	Y	N	
<i Profession> = student, pensioner	-	-	-	Y	
<i Time of day> = afternoons	-	Y	N	Y	
<o admission charge> = 10 euros				■	■
<o admission charge> = 6 euros		■	■		
<o admission charge> = 0 euros	■				
<o admission charge> = reduction of 2 euros		■		■	

G-1.5.3 Formulating Conditions and Actions

Conditions and actions need to be entered in a certain syntax.

The names need to be put in angle brackets. A letter for the type and a space must be entered after the opening angle brackets. If the name's text uses a similar syntax, conflicts may occur upon verification. Parentheses and letters for the type are only shown in windows which aren't locked if you have made settings so this can happen (see chapter G-1.5.4, "Making Angle Brackets Visible", page G-42).

A condition row must be inserted into the decision table for each flow which flows into a process or a bar (see chapter B-4.4.3, "Inserting Rows and Columns", page B-120). The name of an inflowing data flow or control flow must be included in the conditions' text respectively (`<i flow>`).

Formulating Conditions

How to proceed

To formulate a condition:

- » Lock the decision table if necessary.
- » If necessary, insert a row in the condition part (see page B-120).
- » Click in the first cell in the row and select `Edit>Properties....`
The dialog box of the same name appears.
All flows which flow into the decision table's process or bar are listed in the drop-down list `Name`.
- » Select a flow from the drop-down list `Name` and expand the condition's formulation in the field `Name`.
- » Close the dialog box with `[OK]`.
The condition is inserted in angle brackets in the cell. It is identified by the code letter `i`.

You need to have made various basic settings before you can formulate rules:

- Enter the rules using fixed values (e.g. `Y/N`) or using any values? Make this setting for the current table.
- If you use fixed values, enter which values should be shown for activated, deactivated and irrelevant events. The `Y`, `N` and `-` are values as standard. You only need to make this setting once for all SA or RT decision tables.

Formulating Rules

How to proceed

If you want to define other values for activated, deactivated and irrelevant events:

- » Select **Extras>Options...**
The dialog box of the same name appears.
The current values are shown in the list box **Text list** in the **Event Value On**, **Event Value Off** and **Event Value Irrelevant** entries.
- » Select the entry in the list whose value you want to change.
The value is shown in the field below the list.
- » Enter the desired value and click on **[Set]**.
- » Repeat this step with all values which you want to change.
- » Then click on **[Apply]** and close the dialog box with **[OK]**.

How to proceed

To formulate a rule:

- » Select the cells which you want to formulate a rule for and select **Edit>Properties...**
The dialog box of the same name appears.
If you have preconfigured fixed values for the table, the values are shown in a list. If you have permitted any values, a field **Name** is displayed.
- » Select the desired values from the list or enter the desired value in the field **Name**.
- » Close the dialog box with **[OK]**.

Formulating Actions

You need to insert a action row in the action part of the RT decision table for each condition which you formulated in the condition part (see chapter B-4.4.3, "Inserting Rows and Columns", page B-120).

How to proceed

To formulate an action:

- » Lock the decision table if necessary.
- » If necessary, insert a row in the action part (see page B-120).
- » Click in the first cell in the row and select **Edit>Properties...**
The dialog box of the same name appears.
Only one drop-down list **Name** is displayed for SA decision tables. The various action types are also listed in the group box **Type** for RT decision tables. These options control the contents of the drop-down list **Name**.

- » Set the desired action type using the options in the group box **Type** (only for RT decision tables.
 All actions which exist for the current type are listed in the drop-down list **Name**.
 Different identifications are permissible to a certain extent for actions in SA and RT decision tables:

Action	SA	RT
Data flow which flows out to the process or bar	<o flow>	<o flow>
No action No action is listed if one of the corresponding rules is fulfilled. This entry can only occur once in the action part. No other action can be assigned to a rule which occurs due to assignment to this action. The noop text can be used as many times as required by cannot be empty.	<n noop>	<n noop>
Events in the Assigned Sate Diagram		<e event>
Action in the assigned process activation table		<a action>
Activating one of the diagram's processes		<p process>

- » Select an action from the drop-down list **Name** and expand the action's formulation in the field **Name**.
 » Close the dialog box with **[OK]**.
 The action is inserted in angle brackets in the cell. It is identified by the valid code letter respectively.

You can mark whether the current action is carried out or not in the numbered column in the action part. If you do not mark the columns, you can also mark the **else** column.

Various basic settings also need to be initially made here:

- Hide or show the **else** column.
- Set how the cells which the actions will be shown or hidden in are to be marked.
- Set the column title for the **else** column (see page B-121).

Activating and Deactivating Actions

How to proceed

To show the `else` column:

- » Lock the table if necessary.
- » Click in the background of the table window and select `Edit>Properties....`
The dialog box of the same name appears.
- » Select the tab `Properties`.
- » Activate the check box `ELSE column` and close the dialog box with `[OK]`.

G-1.5.4 Making Angle Brackets Visible

The entries are fully displayed in the condition and action parts as standard, i.e. with angle brackets and code letters. You can deactivate this display.

Note **If you deactivate the display of angle brackets, this is only visible in unlocked tables. The parentheses and code letters are always visible in locked tables.**

How to proceed

To change the display:

- » Select `Extras>Options....`
The dialog box of the same name appears.
- » To show angle brackets, activate the check box `Cell type` in the group box `Layout mode`. To hide the parentheses, deactivate the check box.
- » Click on `[Apply]` and close the dialog box with `[Close]`.

G-2 Using Real-Time Extension of Structured Analysis

Traditional structured analysis enables you to describe extensive and complex actions. However, if the system's performance changes due to outside influences or if the system needs to actively control the outside world, then the SA method in the traditional form is no longer suitable.

The so-called real-time extension was developed especially for these cases. Innovator supports real-time extension according to Hatley/Pirbhai. Structured analysis with real-time extension is often called SA/RT. RT stands for real-time.

This extension enables control information to be entered into the flow diagram. A control process is responsible for application flows in a diagram. A control process' specification determines:

- Which events will be acted on
- Which system transitions are possible
- Which activities are activated or deactivated in the system

There are three resources available to you for viewing real-time extension:

- **RT decision tables** allow for the specification of real-time characteristics in SA diagrams in relation with state diagrams and activation tables. To do this, input signals at the bar in the diagram are converted into process activation tables and out flowing control flows (see chapter G-2.1, "Combining Flows in RT Decision Tables", page G-43).
- An RT decision table follows the **state diagram** (see chapter G-2.2, "Modeling System Behavior with State Diagrams", page G-45).
- The **process activation table** forms the end (see chapter G-2.3, "Creating Process Activation Tables for a Bar", page G-50).

Each of these resources used for displays can be left out during specification; it is also possible to omit any two.

Resources for Real-Time Extension

G-2.1 Combining Flows in RT Decision Tables

RT decision tables allow the control specifications of SA diagrams to be determined relating to state diagrams and process activation tables. Assignment to the SA diagram takes place using name similarity.

You can use RT decision tables to set which external influences influence your system and which reactions should be resolved in the system by these respectively. Read what structure decision tables have in Innovator and which rules you should be aware of in chapter G-1.5, "Decision Tables", page G-34.

G-2.1.1 Creating RT Decision Tables

You can create an RT decision table using a bar in an SA diagram or in the model browser.

If you want to create an RT decision table using **an SA diagram**, you will need the `bar` model element. The bar provides an interface between structured analysis and real-time extension.

Note **Only one RT decision table can be created per diagram, as the bar is identified using the current diagram. Each bar in the diagram represents this control specification.**
If an RT decision table already exists for a diagram, you cannot create other tables.

How to proceed

To create an RT decision table:

- » Jump to the SA diagram which you want to create the table for.
- » If necessary, create a bar (see chapter G-1.2.2, "Creating, Adding and Renaming Model Elements", page G-7).
- » Select the bar and select [Engineering>Create Decision Table](#).
The table is created and opened.

Read how to formulate conditions and actions in chapter G-1.5, "Decision Tables", page G-34. Read how to work with tables in chapter B-4.4, page B-118.

G-2.1.2 Opening RT Decision Tables

You can open an RT decision table using the SA diagram which contains the bar or via the model browser.

Opening Tables Via Diagrams

The diagram must not be locked for you to be able open the corresponding RT decision table via the **SA diagram**. No model elements (except for bars) or flows can be selected.

How to proceed

To open the RT decision table:

- » Select a bar.
 - » Select the [Jump>Reference...](#) and the [RT decision table](#) entry in the dialog box.
 - » Close the dialog box with [\[Jump\]](#).
- The RT decision table is opened.

The RT decision tables are stored in the package which the SA diagram and bar are in as standard in the model browser. If you have assigned RT decision tables another package, you can find this using the search function (see chapter B-3, page B-81).

Opening Tables Via the Model Browser

How to proceed

To open an RT decision table via the model browser:

- » Jump to the model browser and open the package which contains the table.
 - » Click on the entry which you want to open and select [Element>Open](#).
- The RT decision table is opened.

Read how to formulate conditions and actions in chapter G-1.5, "Decision Tables", page G-34. Read how to work with tables in chapter B-4.4.4, "Changing Column Titles and Cell Contents", page B-121.

G-2.2 Modeling System Behavior with State Diagrams

State diagrams are part of real-time system modeling defined by the Hatley/Pirbhai method.

They describe a finite state machine which is made up of various states. The machine only ever has one state at one given time. State transitions are executed and actions resolved as a result of external signals (events).

A state diagram, along with the RT decision table and the process activation table, represents the control process specifications in an SA diagram. Assignment to the SA diagram takes place using name similarity.

You can use the following elements to create a state diagram in the diagram editor:

- States
- Transitions

Getting to Know Elements of State Diagrams

- Events/actions

G-2.2.1 Creating State Diagrams

You can create a state diagram using a bar in an SA diagram. The diagram must not be locked.

Note You can only create one state diagram per diagram, as the bar is identified using the current diagram.

If a bar already has a state diagram, another one cannot be created.

How to proceed

To create a state diagram:

- » Jump to the SA diagram which you want to create the state diagram for.
- » If necessary, create a bar (see chapter G-1.2.2, "Creating, Adding and Renaming Model Elements", page G-7).
- » Select the bar and select [Engineering>Create State Diagram](#).
A new window containing the bar's state diagram is opened. The diagram contains the name of the respective SA diagram.

G-2.2.2 Specifying Possible States of the SA Diagram in the State Diagram

States are the only model elements which you can create in a state diagram.

State This element corresponds to a state which a system can exist in. The initial state and the normal state can be distinguished between:

The normal state is shown in the rectangular frame. The specific name of the state is written inside the frame.

Initial states are shown as small circles.

Read how to delete states in chapter B-2.5.3, page B-64.

How to proceed

The state diagram must be locked during all of the following tasks.

- » If necessary, lock the state diagram.

If you create a state, decide whether you want to create an initial state or a normal state.

Creating States

How to proceed

To create a state in a state diagram:

- » If you want to create the state in a certain position, click on this position.
 - » Select **Insert>State** or **Insert>Init State**.
- A state is created. A normal state has a default name.

You can change a state's name and type at a later stage.

Modifying States

If the state should be converted into an initial state it cannot have any inflowing transitions.

Note

How to proceed

To modify a state:

- » Select the state (state or initial state) that you want to modify. Ensure that no other element is selected.
 - » Select **Edit>Properties...**
- The dialog box of the same name appears. The option button **Initial state** is grayed out if you selected a state which has a transition flowing into it.
- » Enter another name in the field **Name**.
 - » Select another type—if this is possible.
 - » Close the dialog box with **[OK]**.
- The changes are visible in the diagram.

G-2.2.3 Inserting State Transitions

The directed relationships between the individual states stand for permissible transitions and are called transitions. Transitions are specified using an initiating event and an action to be implemented. Action specification can also be omitted.

A transition is shown as a straight line between two states or as a line which only ends at one state (initial state). This sort of a line is made up of at least one straight line, but can consist of many more.

You can set whether the event should concern a signal from the SA diagram or an action from the RT decision table:

- **SA signal:** A control flow which flows into an SA diagram's bar

- **ET action:** An event which is created in the RT decision table

For actions, you can choose between SA signals, SA activations and PAT actions:

- **SA signal:** A control flow which flows out of a bar
- **SA activation:** Activating/deactivating an activity in the dataflow diagram
- **PAT action:** An action which is specified in greater detail in the process activation table

How to proceed

The state diagram must be locked during all of the following tasks.

- » If necessary, lock the state diagram.

Creating Transitions

You need to enter at least two states when creating a transition. If you only enter one state, the transitions flows into this state as well as out of it.

How to proceed

To create a transition:

- » Select the source state.
- » If necessary, select the target state.
- » Select [Insert>Transition](#).

Changing Transitions

You can also change a transition at a later time. The following changes are possible:

- Reversing the direction
- Defining another state as the source or target state (see page B-45)

Read how to change events and actions in chapter G-2.2.4, page G-49.

Changing the Direction

When creating the transition, you specified the direction through the order in which you selected the states. You can reverse the direction of the transition between two normal states.

How to proceed

To reverse the direction of the transition:

- » Select the transition and select [Edit>Properties...](#)
The dialog box of the same name appears.

- » Activate the check box **Invert direction** and close the dialog box with **[OK]**.

G-2.2.4 Changing Events or Actions

An event and, if applicable, an action trigger a state transition.

Actions can e.g. be the out flowing flow of a bar or the activation/deactivation of a process in the related SA diagram.

The names on the transitions show the state diagram's interfaces to the control specification's other components, as well as to the flow diagram.

One of the following objects needs to exist in an event's name for state diagrams:

Events

- Inflowing control flow at the bar (**<i flow>**)
- Event in the assigned RT decision table (**<e event>**)

The following elements can be used when formulating actions:

Action

- Action in the assigned process activation table (**<a action>**)
- Out flowing control flow at the bar (**<o flow>**)
- Activating one of the diagram's processes (**<p process>**)
- No action (**<n noop>**)

The names need to be put in angle brackets. A letter for the type and a space must be entered after the opening angle brackets. If the name's text uses a similar syntax, conflicts may occur upon verification. Parentheses and letters for the type are only shown in windows which aren't locked if you have made settings via **Extras>Options... so this can happen.**

Note

You can specify transitions using events and actions.

Entering Events or Actions

How to proceed

To set an action or an event:

- » Select the transition and select **Edit>Properties...**
The dialog box of the same name appears.
- » Set whether the transition is resolved by an SA signal or by the decision table's action using the alternatives in the group box **Event**.
If you enter a new event you do not need to enter the code letters and the angle brackets (see page G-49). They are automatically added.

- » Enter an event in the field **Event** or select the desired event from the drop-down list.
You can limit the drop-down list to SA signals, SA activations or actions in process activation tables using the junctions in the group box **Action**.
- » Using the junctions in the group box **Action**, set which kind of action you want to be resolved by an event.
If you enter a new action you do not need to enter the code letters and the angle brackets (see page G-49). They are automatically added.
- » Enter an action in the field **Action** or select the desired action from the drop-down list.
- » Make the desired changes and close the dialog box with **[OK]**.

G-2.3 Creating Process Activation Tables for a Bar

The effects which certain events have on the processes in the diagram are displayed in the process activation table (PA table).

Incoming signals are converted into process activations and output signals in the process activation table.

You can use a process activation table for specifying control processes, in addition to the RT decision table and state diagram. Assignment to the SA diagram takes place using name similarity.

G-2.3.1 Creating and Opening Tables

If you want to create a process activation table using **an SA diagram**, you will need the **bar** model element. The bar provides an interface between structured analysis and real-time extension.

Note **Only one process activation table can be created per diagram, as the bar is identified using the current diagram.**

If a bar already has a process activation table, another one cannot be created.

How to proceed

To create a process activation table:

- » Jump to the SA diagram which you want to create the PA table for.

- » If necessary, create a bar (see chapter G-1.2.2, "Creating, Adding and Renaming Model Elements", page G-7).
- » Select the bar and select [Engineering>Create Process Activation Table](#).
A new window containing the marked bar's process activation table is opened.

The diagram must not be locked for you to be able open the corresponding process activation table via the **SA diagram**. No model elements (except for bars) or flows can be selected.

Opening Tables Via Diagrams

How to proceed

To open the process activation table:

- » Select a bar.
- » Select the [Jump>Reference...](#) and the [process activation table](#) entry in the dialog box.
The process activation table is opened.

The process activation tables are stored in the package which the SA diagram and bar are in as standard in the model browser. If you have assigned process activation tables to another package, you can find this using the search function (see chapter B-3, page B-81).

Opening Tables Via the Model Browser

How to proceed

To open a process activation table via the model browser:

- » Jump to the model browser and open the package which contains the table.
- » Click on the entry which you want to open and select [Element>Open](#).
The process activation table is opened.

Read how to work with tables in chapter B-4.4, page B-118.

G-2.3.2 Showing and Hiding Column Groups

The process activation table consists of three column groups as standard:

- The first column group only consists of one column which lists the input signals. You need to insert one of the following objects in this column's cells:
 - Inflowing control flow at the bar ([<i flow>](#))
 - Action in the assigned RT decision table ([<a action>](#))

- Action in the assigned state diagram
(`<a action>`)
- You can create a column for each process which should be activated in the second column group called `process group`. The name of a process must exist in each column (without angle brackets).
- The output signals are listed column by column in the last column group, `Output group`. This means that each column must contain the name of a control flow which flows out of the diagram from the bar (without parentheses). You can label the cells as required, as with processes.

You can hide one of the column groups `Process Groups` or `Output Groups`.

Attention **If you hide a column group with contents, this contents is lost. If you show the column again it is empty.**

How to proceed

To hide a column group:

- » Lock the table window.
- » Click in the background of the table window and select `Edit>Properties...`
The dialog box of the same name appears.
- » Select the tab `Properties`.
- » Deactivate the check box for the column group you want to hide in the group box `Sections` and close the dialog box with `[OK]`.

Read how to work with tables in chapter B-4.4, page B-118.

G-2.3.3 Defining Input and Output Signals for Process Activation

You can use the process activation table to set under which circumstances a process in the SA diagram should be carried out or ignored.

If you have already specified an RT decision table or a state diagram you can use their actions as input for the process activation.

Read how to change column titles in chapter B-4.4.4, page B-121.

Creating Process Columns

To set which process should be controlled using the process activation table, create a column for each relevant process. The PAT section `Prozess-gruppe` must be available for this (see chapter G-2.3.2, page G-51).

How to proceed

To enter a process:

- » If necessary, insert a column in the **Prozessgruppe** section (see chapter B-4.4.3, "Inserting Rows and Columns", page B-120).
- » Click on the column's column title (not the group title) and select **Edit>Properties...**
The dialog box of the same name appears. All processes of the related SA diagram are shown in the drop-down list **Name**.
- » Select a process or enter a name.

If your process names are too long, this can make the table extremely wide and unclear. Read how to reduce the element's labeling in chapter B-2.4.1, "Learning Wrapping Rules for Element Names in Models", page B-55.

Note

- » Close the dialog box with **[OK]**.
A new column with the corresponding column title is created within the section.

To set which effects should be resolved using activation of a process, create a column for each flow which flows out of the bar. The **Output Group** section must be available for this (see chapter G-2.3.2, page G-51).

Creating Output Columns

How to proceed

To specify the effects:

- » If necessary, insert a column in the **Output Group** section (see chapter B-4.4.3, "Inserting Rows and Columns", page B-120).
- » Click on the column's column title (not the group title) and select **Edit>Properties...**
The dialog box of the same name appears. All flows which flow out of the bar in the SA diagram are shown in the drop-down list **Name**.
- » Select one of the flows or enter a name.
- » Close the dialog box with **[OK]**.

Create a row in the first column for each input which influences the processes. This input can be:

Creating Input Rows

- A control flow which flows into a bar
- An action which is defined in the bar's RT decision table
- An action which is defined in the bar's state diagram

The name of an inflowing control flow (`<i flow>`) or an action (`<a action>`) must be included in the input's text respectively.

Note **The names need to be put in angle brackets. A letter for the type and a space must be entered after the opening angle brackets. If the name's text uses a similar syntax, conflicts may occur upon verification. Parentheses and letters for the type are only shown in windows which aren't locked if you have made settings so this can happen (see chapter G-1.5.4, "Making Angle Brackets Visible", page G-42).**

How to proceed

To specify the input row:

- » If necessary, insert a row (see chapter B-4.4.3, page B-120).
- » Click in the row's title and select `Edit>Properties....`
The dialog box of the same name appears.
- » Select whether you want to specify a control flow (`Inflow/outflow at bar`), an action from the RT decision table (`Action from decision table`) or an action from the state diagram (`Action from state transition`) in the group box `Type`.
The SA diagram's (inflowing) control flows or the ones in the RT decision table or in the actions specified in the state diagram are listed in the drop-down list `Name`.
- » Select a control flows or action or enter a name.
The code letters `i` or `a` and the angle brackets are automatically added.
- » Close the dialog box with `[OK]`.
The first column contains the name entered—in angle brackets and with the code letter.

Activating Processes

You can set how the process should be influenced in the `Prozessgruppe` section's cells.

Various processes can be activated or deactivated using an input signal. You can enter any text in the cells in the process columns. e.g. you can mark the processes which should be activated with an `X`. The the order of the activation is important, numbering should also be considered.

How to proceed

To specify the activation:

- » Click in one of the `Input/Process` cells and select `Edit>Properties....`
The dialog box of the same name appears.

- » Enter a description for the process activation in the field **Name**.
- » Close the dialog box with **[OK]**.

The activation is transferred to the cell.

G-2.4 Creating Time Requirements Tables for a Terminator

Time requirement tables are used in structured analysis in context diagrams to meet assertions via the answer patterns of system processes. Target values can be set for the period of time from inputting a terminator in the system process to the system response. System response means output to a terminator. Input and output terminators do not need to be identical.

A time requirements table is always assigned to the terminator which results from an input. The table consists of five columns:

- **Input**
Any text can be entered. The name of a flow which leads from the terminator to a system process must be included in this text (**<i flow>**).
- **Event**
This text provides more information about the input signal. It can also be used for case differentiation for the signal's various states.
- **Output**
Any text can be entered. The name of a flow which leads from a system process to the terminator must be included in this text (**<o flow>**).
- **Event**
This text provides more information about the output signal.
- **Time**
The response time can be specified in this column. The maximum response time is usually entered. A special format is not required when entering.

A robot come with a remote control so that it can be turned off or activated. The robot's current state (active or inactive) should be shown on the display. How to turn the robot off should also be mentioned in a protocol. Information should appear on the display within half a second (turning off) or three seconds (activation). The permissible maximum time for the protocol is 10 seconds.

Example:

The Time requirements table then looks like this:

Input	Event	Output	Event	Time
<i CMD>	On	<o mode> (Display)	green	3 sec
<i CMD>	Off	<o mode> (Display)	red	0.5 sec
<i CMD>	Off	<o activity> (Recorder)	turning off	10 sec

How to proceed when editing a time requirements table is similar to how you would edit a process activation table. Read how to edit a table in the following chapters.

- **Changing column titles:** chapter "Changing Column Titles and Cell Contents"
- **Creating input rows:** section "Creating Input Rows", page G-53
- **Creating outputs:** section "Creating Output Columns", page G-53

Creating Tables

If you want to create a time requirements table using an **SA diagram**, you will need the **Terminator** model element. These are only available in the context diagram as standard (see chapter G-1.2.1, "Limiting the Use of Model Elements in Diagrams", page G-5).

Note If a time requirements table already exists for a terminator, another one cannot be created.

How to proceed

To create a time requirements table:

- » Jump to the context diagram which you want to create the real-time extension for.
- » If necessary, create a terminator (see chapter G-1.2.2, "Creating, Adding and Renaming Model Elements", page G-7).
- » Select the terminator and select **Engineering>Create Time Requirements Table**.

A new window containing the marked terminator's time requirements table is opened.

Read how to work with tables in chapter B-4.4.4, "Changing Column Titles and Cell Contents", page B-121.

G-3 Displaying Call Hierarchies Using Operation Diagrams

Structured Design (SD) bridges the gap between technology independent analysis and actual implementation. Structured design introduces technical conditions and determines a system's outline for technical purposes, thereby displaying the contents' planning of the implementation. It is therefore the content planning of the implementation.

The outline is graphically displayed in the form of data areas, operations and modules, as well as interfaces to these elements. Innovator supports the method defined by Yourdon/Constantine for operation and data level descriptions. In addition, operations can be grouped to modules and a module hierarchy can be graphically displayed

A very general grouping potential is also available with the package concept. You can arrange your system's design elements in any hierarchy and, in doing so, display certain data between subsystems in a clear way.

The call hierarchy of operations is modeled in operation diagrams. Which parameters should be assigned for each operation and data field can also be specified. You can create and edit operation diagrams using the SD diagram editor.

You can use the following elements to create an operation diagram:

- Operation (see page G-58)
 - Internal operation
 - External operation
- Data (see page G-58)
 - Internal data
 - External data
- Connector (see page G-67)
- Call (see page G-71)
- Parameter (see page G-74)
 - Data parameter
 - Control parameter
 - Hybrid parameter

Getting to Know Elements of Operation Diagrams

G-3.1 Creating and Renaming Operation Diagrams

Create the operation diagram in the model browser. This is where you can initially decide which package the new operation diagram should be stored in. Read about packages in chapter A-3.1, page A-36.

- Other Information**
- chapter B-2.2.2, "Creating and Opening Diagrams", page B-34
 - chapter B-2.3.1, "Changing Names, Relationships and Declarations", page B-49
 - chapter B-2.5, "Removing and Deleting Elements", page B-61
 - chapter B-4, "Working in Diagrams or Tables", page B-99

G-3.2 Working with Operations and Data Areas

An operation diagram illustrates your system's call structure.

SD operations represent a programming language's operations. Operations can be the starting or end points of call links. An operation differentiates between external and internal operations.

SD data represents data areas of a programming language, such as e.g. variable declarations. They can only exist at a call link's end point, not at the start point. A data area differentiates between external and internal data areas.

Internal Operations An operation is displayed as internal operation if it still needs to be created for this project and is not already predefined. One or more operations to be edited in a module diagram can be assigned to a module. Operations are shown in rectangles. The rectangle contains the name of the operation. The structured source code of an operation can be entered for each operation. Operation calls, input and output parameters are all automatically taken into consideration. A process, a terminator or a structured analysis (SA) data store can be assigned to an operation.

External Operations External operations represent operations which already exist (e.g. in existing libraries) and merely have to be integrated into the new project. External operations are shown in rectangles with two additional vertical lines. The rectangle contains the name of the operation. External operations cannot be displayed, i.e. their source code cannot be called or edited.

A data area is displayed as internal data if it still needs to be created for this project and is not already predefined. One or more data areas can be assigned to a module, regardless of whether this module also has operations assigned to it. Data areas are shown in hexagons. The name of the data area is written in these hexagons. A source code can be entered for each data area which e.g. can be made up of variable declarations and type definitions, but not up of operations. A data area can be assigned to a store from SA.

Internal Data

External data represents data areas which already exist (e.g. in existing libraries) and merely have to be integrated into the new project. External data is shown in a hexagon with two additional vertical lines. The hexagon contains the name of the data area. External data cannot be displayed, i.e. their source code cannot be called or edited.

External Data

G-3.2.1 Creating, Adding and Changing Operations or Data Areas

There are many ways to create operations and data areas:

- In the operation diagram (see page B-39)
- In the model browser in the desired package (see page B-38)

You can also use an element which you have created in a diagram in another diagram (for information about how to add it, see page G-60). You can decide whether you only want to make this element visible or whether all of the links to this element in other diagrams should also be transferred into the current diagram. Read what to be aware of when adding elements in chapter B-2.1.2, "Logical and Graphic Existences of Model Elements", page B-22.

You can also copy operations and data areas (see page G-60).

- chapter B-2.3.1, "Changing Names, Relationships and Declarations", page B-49
- chapter G-3.2.2, "Enabling Various Main Operations in the Model", page G-61
- chapter B-2.5, "Removing and Deleting Elements", page B-61

Other Information

Adding Elements If operations or data areas are created once, they can be added to any number of operation diagrams and therefore display special aspects (see page B-22).

How to proceed

If you want to add an element which already exists somewhere else:

- » Jump to the operation diagram and lock it.
- » Select **Insert>[Internal|External] [Operation|Data]>Add Existing...**
The dialog box of the same name appears.
All elements of the given type which were already created in the model and do not already exist in the current diagram are listed in the drop-down list **Name**.
- » Select the element which you want to add to the current diagram from the drop-down list **Name**.
- » Using the junctions in the group box **Maintenance**, set which maintenance status the element should have in the current diagram. For information about the maintenance status, read chapter B-2.1.3, page B-24.
- » Click on **[Apply]** and close the dialog box with **[Close]**.
Another graphic existence of the element is visible in the diagram.

Copying and Inserting Diagram Elements

You can copy the following elements:

- **Operations and data areas**

If you copy operations and data areas, you can only reinsert these in one other operation diagram.

In doing so you create another graphic existence of the original module, i.e. you add the element into another diagram.

Operations, data areas and calls which are not already available in the diagram are not only inserted once. This constraint does not apply for parameters and comments.

The maintenance state of inserted elements is adopted from the original diagram.

- **Calls**

Copied calls can only be reinserted into other diagrams if the target element from the original diagram already exists there and a source element is selected.

If a call is inserted, its parameter is also inserted.

- **Parameter**

You can reinsert copied parameters into any number of calls.

- **Comments**

There are no constraints when copying comments.

You can select, copy and reinsert the listed elements individually or at the same time. You can select other diagram elements at the same time, but these will not be added to the buffer.

How to proceed

The source operation diagram cannot be locked when copying elements, but target operation diagrams, on the other hand, can be:

- » Select the elements which you want to copy and select **Edit>Copy**.
The elements are copied in the buffer.
- » To reinsert the elements, if necessary, jump to the operation diagram where you want to insert the elements and lock it.
 - If you have copied an operation, data or a comment, you can set the position where the copy should be inserted. Click on the desired position.
 - If you have copied a call, the element which has been the call's target element up to now is also the target element for the copy. You can specify a new source code. Select the operation which should be the calling element for the new call.
 - If you have copied a parameter, select the call which you need a copy for.
- » Select **Edit>Paste**.
The elements are visible in the diagram.

G-3.2.2 Enabling Various Main Operations in the Model

Names of operations must be unique throughout the models. This means that you cannot display various C programs each with one **main** operation within the model.

Innovator provides an option for designing various programs in the model without ruining the success of name uniqueness.

If you want to model various **main** operations, proceed as follows:

- Define a so-called main prefix for your model, i.e. a short string which will precede each **main** operation later.
- Create the desired operation diagrams for the various C programs. Instead of the **main** operation, create one operation with a unique name respectively, but enter the main prefix in front of this name.
- When making adjustments using the implementation, Innovator searches for operations which have the main prefix in the name and renames them automatically to **main**.

Procedure

Defining Main Prefixes

You can only set the main prefix if you have set the programming language as **C**.

How to proceed

You require administrator rights to define the main prefix:

- » Jump to the model browser and log-in as administrator (see chapter A-1.2.1, page A-5).
- » In the model browser, select **Model>Configuration>General Settings...**
The dialog box of the same name appears.
- » Select the tab **Implementation**.
- » Ensure that the entry **C** is set in the group box **Language**.
- » Enter the string which you want to precede all **main** operations later in the group box **C** in the field **Main prefix**.
- » Close the dialog box with **[OK]**.
All operations whose names are preceded with the main prefix from now on are automatically renamed to **main** during implementation.

G-3.2.3 Transforming External Operations or Data Areas into Internal Operations or Data Areas

Both operations and data can exist as internal or external (see page G-58). You can transform an element into the respective other type at a later stage.

Note **Type-based assignments of modules are lost upon transformation. Related implementations (e.g. operations, data areas or modules) cannot be locked for project-internal elements.**

How to proceed

To change between external and internal elements:

- » If necessary, lock the diagram.
- » Select the element which you want to transform and lock it.
- » Select **Edit>Properties...**
The dialog box of the same name appears.
- » Select the tab **Properties**.
The current setting is shown in the group box **Object type**.
- » Activate the junction which you want to transform the element in.
- » Close the dialog box with **[OK]**.
The element is transformed.

G-3.2.4 Defining Formal Interfaces of Operations and Data Areas

You can set which formal parameters should be used in an operation or a data area. To do this, specify the name, type, order and—for operations—the direction of the parameter. Also set whether the information concerns a data, control or hybrid parameter (see page G-74).

If you provide an element, which has the formal parameter, as a calling operation or a data area with a call link, the formal parameters are automatically suggested as the current parameters for the call link.

Note

You can use the suggested formal parameters or create them individually.

You can adopt the current parameters which already exist for the current element's calls in the model as formal parameters.

Adopting Current Parameters as Formal Parameters

If formal parameters are already defined for the current element, these are replaced by the current parameter.

Note

How to proceed

For the current parameter for the element's call from the entire model to be suggested as the formal parameter:

- » Click in the dialog box **Edit/Properties** (see page G-63) in the tab **Definition** on **[Default]**.
All parameters which already exist as the element's current parameters are inserted into the list which is labeled with the current element name.
- » Close the dialog box with **[OK]**.

You can define as many parameters as required in the current element.

Creating Formal Parameters

How to proceed

To define formal parameters in an operation or a data area:

- » Select the element which you want to define the parameters for and lock it.
- » Select **Edit>Properties...**
The dialog box of the same name appears.

- » Select the tab **Definition**.

All formal parameters which are already defined for the element are listed in the group box **Parameters**. The parameters are labeled as follows:

Type	Abbreviation
Data	D
Control	C
Hybrid	H

Direction	Depiction
Input	-->
Output	<--
Input/output	<-->
Return	<<<

You can set the direction of the parameter for operations. This is not possible for data areas.

- » If the selected element is an operation, use the junctions in the group box **Direction** to set the direction of the new parameter.

The field **Name** is grayed out if you activate the **Return** radio button.

- » If you have selected anything other than option button **Return**, enter the parameter's name in the field **Name**.

The DD entries and types of all parameters which are not used in the data dictionary, i.e. only used in the structured design, are offered in the drop-down list **Datentyp**.

Note If you create an implementation from the operation, the * asterisk is automatically added for the pointer for the **Output** and **Input/output** directions. Only enter an asterisk if you also need one for this pointer.

- » Enter the parameter's type in the field **Datentyp** or select one from the drop-down list.
- » Use the junctions in the group box **Type** to set whether the information concerns a data, control or hybrid parameter.
- » Click on **[Add]** to include the parameter in the list.
The parameter appears in the list box **Parameters**.
- » Create all parameters for the current element in this way.
- » Finally, close the dialog box with **[OK]**.

Not only types and amounts of the formal parameter are relevant when defining the formal interface, the order also plays a role. You can change the order of the parameters in the list box **Parameters**.

Changing the Order of Parameters

Only one return parameter can exist for each element. This is always listed first in the list. You cannot alter this position.

Note

How to proceed

To change the order of the formal parameters:

- » Select the formal parameter in the dialog box **Edit/Properties** (see page G-63) in the list box **Parameters** whose position you want to change. Ensure that this is not a return parameter.
- » Move the parameter to the desired position using the **[Up]** or **[Down]** buttons.
- » Close the dialog box with **[OK]**.

You can change or delete a parameter's properties at a later stage.

Changing and Deleting Parameters

Current parameters which were automatically created from the formal parameters are not initially influenced by these changes. You can, however, transfer the changes to the current parameters at a later stage (see page G-75).

Note

How to proceed

To change an element's formal parameter:

- » Select the formal parameter in the dialog box **Edit/Properties** (see page G-63) in the tab **Definition** in the list box **Parameters** whose properties you want to change.
The settings on the right-hand side of the window align themselves with the parameter's current properties.
- » Make the desired changes and click on **[Change]**.
The changes are shown in the list.

How to proceed

To delete an element's formal parameter:

- » Select the formal parameter in the dialog box **Edit/Properties** (see page G-63) in the list box **Parameters** which you want to delete.
The settings on the right-hand side of the window align themselves with the parameter's current properties.
- » Click on **[Delete]**.
The parameter is deleted from the list.
- » Close the dialog box with **[OK]**.

Enabling Variable Arguments

If the number of arguments cannot be predicted, enter this as a variable for the current element. The parameters which you have defined in the list box **Parameters** still apply. However, it is also possible to use other arguments.

How to proceed

To permit variable elements in the current element:

- » Activate the check box **Variable arguments** in the dialog box **Edit/Properties** (see page G-63).
- » Close the dialog box with **[OK]**.

G-3.2.5 Assigning SD Elements to SA Elements

You can assign one or more elements from the operation diagram to (no more than) one element from the structured analysis. This ensures that processes, terminators and stores which you have identified in the analysis are also taken into consideration in the design.

Note **You can only assign a data area to a store from the SA.**

You can assign any number of SD elements to SA elements. One-to-one mapping is not necessary.

You can either assign SD elements in the model browser or in the operation diagram.

How to proceed

To assign an SD element to an SA element:

- » Select the elements in the operation diagram or in the model browser's list of model elements which you want to assign to an SA element and lock it.
- » Select **Engineering>Assign to....**
The dialog box of the same name appears.
- » If applicable, select the tab **SA Assignment**.
If a data area is stored in the selected SD elements, you can only assign this to one SA store.
- » Activate the junction in the group box **Type** for the SA element type which you want to assign to the selected elements.
All SA elements of the corresponding element type which already exist in the model are listed in the list box **Object**. If the SD elements are already assigned to one of the SA elements, the appropriate entry is marked, the **[Assign]** button is grayed out.
- » Select the desired element in the list box **Object** and click on **[Assign]**.
- » Close the dialog box with **[Close]**.

G-3.3 Making Diagrams with Connectors Clearer

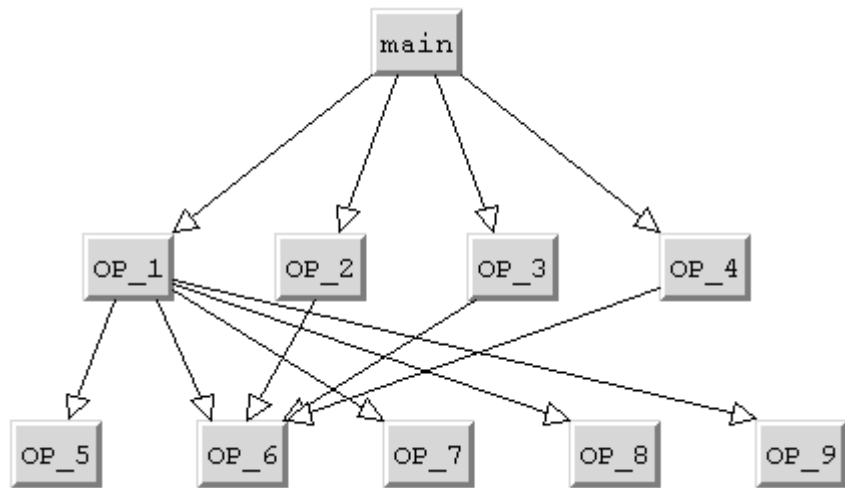
Diagrams which contain a operation which is called by various elements can quickly become extremely unclear, as it is practically impossible for calls not to cross over each other.

A connector serves as a graphical aid for displaying the continuation of an operation structure. It improves the visual representation of a diagram.

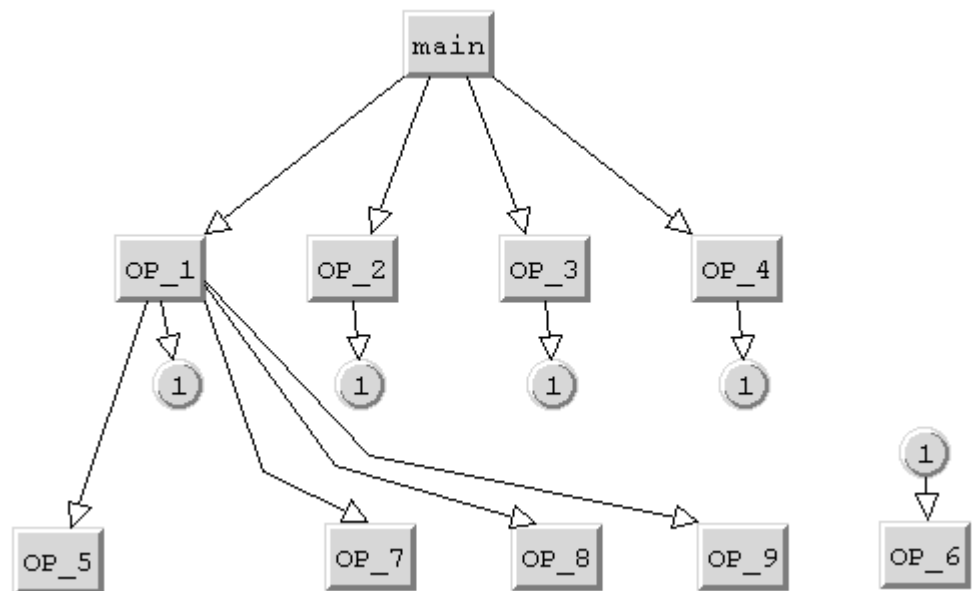
Connectors are shown as circles. The name of the connector is written in the circle.

A connector represents an operation (or this operation's call). If you want to display a call on an operation which has a connector, a connector is created between the operations instead of a relationship line. This way you can avoid overlapping of calls and achieve a clear layout for your diagram.

Connector



Displaying Without Connectors



Displaying With Connectors

Start Connectors and Target Connectors

Start and target connectors are distinguished between. Start connectors show the start of a call structure. There is always one calling operation attached to a start connector. Target connectors show a break or continuation of the call structure to another position. Target connectors are attached to precisely one operation (one module).

A start connector can also exist even if no target connectors exist. In this case, the connector's call links do not have an arrow point.

Various applicable target connects which have the same name can exist for one start connector. i.e.: Various places in the module or operation diagram can refer to the same continuation.

Each start connector, on the other hand, is unique; there can only be one place where this connector is used as a starting point for a module or an operation structure.

G-3.3.1 Creating and Renaming Connectors

You can create a connector in various situations:

A: You already know that an operation is called so often that it should be displayed as a connector. Set which operations you really want to call.

Using this method, you initially only create the start connector. If you create a call for this connector at a later stage, a target connector is created for the calling operation.

B: An operation is already called by another. You only wish to display this call using a connector.

You can use this method to create a start connector which is linked with the calling operation and a target connector which is linked with the calling operation at the same time.

How to proceed

The diagram must be locked during all of the following tasks:

- » Lock the diagram.

Each operation within an operation diagram can be represented by no more than one connector. If a connector already exists for an operation, another one cannot be created.

Creating Connectors for Operations

How to proceed

The operation itself cannot be locked when creating a connector for an operation:

- » Select the operation which you want to create a connector for and select **Insert>Connector**.

A connector is linked with the operation as a small circle (start connector). The relationship line does not have an arrow point as it is not yet called by any other operations.

You can split an operation's call using a connector. Two circles of the same name then appear: One which is linked with the calling operation (target connector) and one which is linked with the called operation (start connector).

Creating Connectors for a Call

If a start connector already exists for the called operation, a new connector is not created for it. The calling operation contains a target connector with the name of the corresponding start connector.

Note

How to proceed

To create a connector for a call:

- » Select the call which you want to split using a connector. Ensure that no connector is already attached to this call.
- » Select **Insert>Connector**.

The call is split. A connector now appears at each free end.

Creating Calls for Existing Start Connectors

If an operation already has a start connector in the current diagram, you can create a call to this connector for each of the other operations and, in doing so, create the desired operation. The call link is then not drawn through right to the operation; a target connector is created which is linked with a short call link instead.

How to proceed

To create a call for a connector:

- » Select the calling operations.
- » Then select the called operation's start connector.
- » Select **Insert>Call**.

A target connector is attached to the calling operation. This has the same name as the start connector.

Renaming Connectors

Connectors are serially numbered within an operation diagram as standard. You can rename connectors at a later stage. It is sufficient to rename just one of the connectors which belong together. All connectors which are linked with this connector are then automatically given this name.

How to proceed

To rename a connector:

- » Select the connector which you want to rename (start or target connector) and select **Edit>Properties...**

The dialog box of the same name appears. The current connector's name is shown in the field **Name**.

- » Enter the new name of the connector in the field **Name**.
- » Close the dialog box with **[OK]**.

The selected connector and all of the connectors linked with it are given the new name.

G-3.3.2 Following Connectors

You can list all the target connectors linked to a start connector and jump to the related operation. You can also jump from a target connector to the start connector it is linked to.

How to proceed

To jump from a start connector to one of its target connector:

- » Select the start connector and select **Jump>Connector**.
The dialog box of the same name appears. All operations which have a target connector from the current start connector are listed in the list box **Befehl**.
- » Select the desired operation in the list box **Befehl** and click on **[OK]**.
The operation is selected.

How to proceed

To jump from a target connector to its start connector:

- » Select the target connector and select **Jump>Connector**.
The operation with the start connector is selected.

G-3.4 Displaying Calls of Operations and Data Areas

An operation diagram illustrates the call hierarchy of operations and data areas. You can link calling operations with called operations or data areas using so-called call links. You can use parameters to display which data is transferred to the called element or which of these the element returns.

A call is shown as a link connecting two operations or modules. The links do not have a name. The link always starts at the calling operation; the link's arrow head points towards the called operation.

Call

You can explicitly create call links yourself or adopt them from the call structure in other operation diagrams.

Note

G-3.4.1 Creating an Operation's Calls

You can use calls to link operations with operations, data areas or connectors.

There are two ways to display calls in the current diagram:

A: Create a call from new, i.e. a relationship didn't exist between the elements involved anywhere in the model up to now.

B: The elements involved are already linked in another diagram. You want to display this relationship in the current diagram.

These calls generally make the elements involved visible using the maintenance status. For information about the maintenance status, read chapter B-2.1.3, page B-24.

If you have checked version objects into your model, inconsistencies may exist in the call structure. If this is the case, adjust the call links of the entire model and, in doing so, complete the missing call links in the current diagram.

Other Information

- Read how to change the call's source or target in section "Assigning a Relationship to a New Model Element", page B-45.
- chapter B-2.5, "Removing and Deleting Elements", page B-61

Creating New Calls

If you create a call as new, the system checks whether a formal parameter is already defined in the called element. These parameters are then shown as a suggestion as the current parameters for the new call.

If you create a call, this call is also incorporated into all diagrams which the elements involved are displayed in, as long as no user-defined maintenance is set for these (see page B-24).

Note **The calling element must be an operation. Data areas can only be a call's target.**

How to proceed

To create a call:

- » If necessary, lock the diagram.
The order in which you select the element involved determines the direction of the call.
- » Select the operation which should be the calling operation.
- » Then select the elements which should be called.

- » Select **Insert>Call**.

The elements are linked by an arrow from the calling to the called elements. If one of the called modules is a connector, an arrow with an extra target connector is created instead of an arrow (see page G-69).

If formal parameters are already defined for a called element, these are displayed as the current parameters for the call.

The name of the called operation appears in the comment in the implementation for the calling operation and vice versa.

If inconsistencies occur regarding calls upon checking-in of version objects, you can adjust the current diagram with all of the model's other operation diagrams. This appends all missing calls between your diagram's elements.

Adjusting Call Links with Operation Diagrams

How to proceed

To append missing calls in the current diagram:

- » If necessary, lock the diagram.
- » Select **Engineering>Adjust>Operation**.
Calls which exist between the same elements in other operation diagrams are entered.
For elements which have their maintenance status set to global, calls are used to add the linked elements in the diagram if necessary.

G-3.4.2 Setting the Order of Calls

Any number of calls can run out of the calling operation. You can set which order these calls should run out of the calling operation.

How to proceed

At least two calls must run out of the operation for you to be able to set the order of the calls:

- » If necessary, lock the diagram.
- » Select the operation which you want to set the order of its calls for. Ensure that this is not a calling operation.
- » Select **Edit>Properties...**
The dialog box of the same name appears.
- » Select the tab **Order**.
All elements which are called from the current operation are listed in the list box **Order**.
- » In the list, click on the entry whose position you want to change and move it to the desired position using the **[Up]** or **[Down]** buttons.

- » Repeat this for all entries whose position you want to change.
- » Finally, close the dialog box with **[OK]**.
If the operation call for the selected operation is generated in the implementation template's comment for the calling operation, the calls appear in the order entered.

G-3.5 Using Current Parameters

Current parameters can be identified by their name and data type. They are shown as little arrows with a circle at the start of the arrow. They are found next to the call links and labeled with the parameter name.

If the arrow points towards the called operation, it is an input parameter of this operation. If the arrow points towards the calling operation, it is an output parameter. If the arrow points in both directions, it is an input and output parameter. Parameters with a double arrow are return parameters. A call link can only have one return parameter.

The arrow always represents a current transfer parameter, never a formal one.

Current parameters can act as data, control or hybrid parameters.

Data Parameters	Data parameters specify internal data which has to be processed. They are shown as empty circles.
------------------------	---

Control Parameters	Control parameters specify certain control elements, such as e.g. switch settings, signals, exception indicators and error indicators. They are shown as filled in circles.
---------------------------	---

Hybrid Parameters	Hybrid parameters can specify both data and control signals e.g. a value in the case of success and an error indicator in the case of an error. They are shown as empty circles with a dot in the middle.
--------------------------	---

G-3.5.1 Creating Current Parameters

Current parameters show the data transfer between the operation diagram's elements. They are shown in the respective calls.

You can create current parameters directly on a call link. However, as current parameters must correspond to the formal parameters of the called element, necessary parameters can also be suggested to you.

If you create a new call link to an element which already has formal parameters, the formal parameters are automatically suggested as the current parameters for the call link.

Note

Read how to delete parameters in chapter B-2.5.3, page B-64.

The call already needs to exist between two elements to be able to create a parameter.

Creating New Parameters

How to proceed

To create a current parameter:

- » If necessary, lock the diagram.
- » Select the call link which you want to create a parameter for and select [Insert>Parameter](#).
An input parameter with a default name is created. You can change the name, type and direction of the parameter if you require.

If formal parameters are already defined for a called element, these are automatically suggested as the current parameters when a call is newly created.

Adopting Formal Parameters as the Current Parameters

You can also make this suggestion at a later stage once the call has been created. Parameter adjustment takes place in all diagrams which the affected call is displayed in.

How to proceed

To use the formal parameters as the current parameters:

- » If necessary, lock the diagram.
- » Select the call you want to display as the current parameters for the target element's formal parameters or select the element which defines the formal interface.
- » Select [Engineering>Adjust>Parameter](#).
As long as formal parameters are defined, they are suggested as the current parameters for the call. Name, type and direction all correspond to the formal interface.

G-3.5.2 Changing Current Parameters

Newly-created parameters are equipped as input parameters with a default name as standard. Current parameters which you have adopted from the formal interface contain properties of the corresponding formal parameter. You can also change the name, type and direction of the parameter at a later stage. This does not influence the formal interface.

How to proceed

To change a current parameter:

- » If necessary, lock the diagram.
- » Select the parameter that you want to change. Ensure that only one parameter is selected.
- » Select **Edit>Properties...**
The dialog box of the same name appears. The settings correspond to the parameter's current properties.
- » Enter the new name of the parameter in the field **Name**.
All types which are already used or are in the data dictionary are listed in the drop-down list **Data type**. You can limit the list to DD types which are already in the include files.
- » If applicable, activate the check box **Implemented**.
- » Enter the parameter's type in the field **Data type** or select one from the drop-down list.
- » Use the radio buttons in the group box **Type** to set whether the information concerns a data, control or hybrid parameter.
If the called element is a data area you cannot enter return as the direction of the parameter. The corresponding option is grayed out in the dialog box.
- » Set the parameter's direction using the junctions in the group box **Direction**.
- » Close the dialog box with **[OK]**.
The altered parameter appears next to the call. The change takes place in all diagrams which the respective call is displayed in.
If necessary, the comments in the calling operation's implementation are adapted to the altered parameter names and data types.

G-3.5.3 Changing the Order of Current Parameters

You can change the order of a call's current parameters. At least two current parameters need to exist for the call.

How to proceed

To change the order of the current parameters:

- » If necessary, lock the diagram.
- » Select the call which you want to change the order of its parameters for.
- » Select **Edit>Properties...**
The dialog box of the same name appears. All current parameters of the call are shown in the list box **Order** in their current order.
- » Select the parameter whose position you want to change in the list.
- » Move the parameter to the desired position using the **[Up]** or **[Down]** buttons.
- » Close the dialog box with **[OK]**.
If the operation call for the selected call is generated in the implementation template's comment for the calling operation, the parameters appear in the order entered.

G-3.5.4 Assigning SD Modules to Elements

You can set which operations should be implemented in which modules in the operation diagram.

The desired module must already be available in the model before you can assign the operation diagram's element to a module.

You can assign various elements to a module at the same time. However, these can only be either all external elements or all internal elements.

If implementations already exist for the elements or the module, these cannot be locked.

Assigning Elements

How to proceed

To assign an element to a module:

- » Select the elements which you want to assign to a module and lock them.
- » Select **Engineering>Assign to...**
The dialog box of the same name appears.
- » Select the tab **SD Assignment**.
All modules which already exist in the model are listed in the list box **Module**. If the elements are already assigned to a module, the corresponding module is selected, the button **Assign** button is grayed out.
- » If necessary, restrict the list by entering a text sample in the field **Filter**.

- » Select the module from the list box **Module** which you want to assign the selected elements to.
- » Click on **[Assign]**.
- » Close the dialog box with **[Close]**.
The selected elements are assigned to the selected module.
The assignment is incorporated into the lifelink if implementations exist for both the module and the element.

Removing Assignments

An element cannot be assigned to more than one module. You can remove the assignment by:

A: Assigning the element to another module (see page G-77)

B: Delete the assignment without adding a substitute

How to proceed

To delete an assignment:

- » Select the element which has an assignment to a module which you want to remove and lock it.
- » Select **Engineering>Assign to....**
The dialog box of the same name appears.
- » Select the tab **SD Assignment**.
The module which the element is assigned to is selected.
- » Click on **[Remove]**.
- » Close the dialog box with **[Close]**.
The assignment of the element to its modules is canceled.
The cancelation is incorporated into the lifelink if implementations exist for both the module and the element.

G-4 Merging Operations in Modules

The call hierarchy relationship of operations at a model level are displayed in module diagrams. You can create and edit using the SD diagram editor.

You can use the following elements to create a module diagram:

- Module (see page G-79)
 - Internal module
 - External module
- Connector: the connector is used in the module diagram in the same way as in the operation diagram. Read how to work with connectors in chapter G-3.3, page G-67.
- Call (see page G-82)

Getting to Know Elements of Module Diagrams

G-4.1 Creating and Renaming Module Diagrams

Create the module diagram in the model browser. This is where you can initially decide which package the new module diagram should be stored in. Read about packages in chapter A-3.1, page A-36.

- chapter B-2.2.2, "Creating and Opening Diagrams", page B-34
- chapter B-2.3.1, "Changing Names, Relationships and Declarations", page B-49
- chapter B-2.5, "Removing and Deleting Elements", page B-61
- chapter B-4, "Working in Diagrams or Tables", page B-99

Other Information

G-4.2 Working with Modules

A module diagram illustrates your system's call structure at a module level.

You can create module diagrams before or after creating operation diagrams with individual function calls.

Module Modules can be the starting or end points of calls. A module is made up of one or several operations which are modified in an operation diagram. A module differentiates between external and internal elements.

Internal Modules A module is displayed as an internal module if it is to be created in this project and is not predefined. Internal modules comprise internal operations. Internal modules are made up of internal operations. Modules are shown in rectangles. The name of the module is written in the rectangle.

External Modules External modules are made up of external operations. External modules are shown in rectangles which also have two additional vertical lines in them. The name of the module is written in the rectangle. External modules cannot be displayed because their source code cannot be edited.

Connectors A connector serves as a graphical aid for displaying the continuation of a call structure. It improves the visual representation of a diagram.
Connectors are used in the module diagram in the same way as in the operation diagram. Read how to work with connectors in chapter G-3.3, "Making Diagrams with Connectors Clearer", page G-67.

G-4.2.1 Creating, Adding and Changing Modules

There are many ways to create a module:

- In the module diagram (see page B-39)
- In the model browser in the desired package (see page B-38)

You can also use a module which you have created in a diagram in another diagram (for information about how to add it, see page G-81). You can decide whether you only want to make this element visible or whether all of links to this element in other diagrams should also be transferred into the current diagram. Read what to be aware of when adding elements in chapter B-2.1.2, "Logical and Graphic Existences of Model Elements", page B-22.

You can also copy modules (see page G-81).

Other Information

- chapter B-2.3.1, "Changing Names, Relationships and Declarations", page B-49
- chapter B-2.5, "Removing and Deleting Elements", page B-61

If modules are created once in the model, they can be added to any number of module diagrams and therefore display special aspects (see page B-22).

Adding Elements

How to proceed

If you want to add a module which already exists somewhere else:

- » Jump to the module diagram and lock it.
- » Select **Insert>Internal Module>Add Existing...** or **Insert>External Module>Add Existing...**
The dialog box of the same name appears.
All modules of the given type which were already created in the model and do not already exist in the current diagram are listed in the drop-down list **Name**.
- » Select the module which you want to add to the current diagram from the drop-down list **Name**.
- » Using the junctions in the group box **Maintenance**, set which maintenance status the element should have in the current diagram. For information about the maintenance status, read chapter B-2.1.3, page B-24.
- » Click on **[Apply]** and close the dialog box with **[Close]**.
Another graphic existence of the model is visible in the diagram.

When copying modules, these can only be reinserted in one other module diagram. In doing so you create another graphic existence of the original module, i.e. you add the element into another diagram (see page G-81). You can individually select, copy and reinsert modules or at the same time.

Copying and Inserting Modules

How to proceed

The diagram does not have to be locked when copying a module:

- » Select the modules which you want to copy and select **Edit>Copy**.
The modules are copied in the buffer.
- » To reinsert the modules, if necessary, jump to the module diagram where you want to insert the modules and lock it.
- » Select **Edit>Paste**.
The modules are visible in the diagram.

G-4.2.2 Transforming External Modules into Internal Modules

Modules can exist as both internal and external modules (see page G-79). You can transform a module into the respective other type at a later stage.

Note **Operations which were assigned to the transformed type can possibly no longer be assigned after the transformation.**
Related implementations (e.g. operations, data areas or modules) cannot be locked for project-internal modules.

How to proceed

To change between external and internal modules:

- » If necessary, lock the diagram.
- » Select the module which you want to transform and lock it.
- » Select **Edit>Properties...**
The dialog box of the same name appears.
- » Select the tab **Properties**.
The current setting is shown in the group box **Object type**.
- » Activate the junction which you want to transform the module in.
- » Close the dialog box with **[OK]**.
The module is transformed.

G-4.3 Displaying Calls of Modules

A module diagram illustrates the call hierarchy of modules. Link calling modules with called modules using so-called call links.

Call A call is shown by a link which connects two operations or modules. The links do not have a name. The link always starts at the calling module; the link's arrow head points towards the called module.

Note **You can explicitly create call links yourself or adopt them from the call structure in other module or operation diagrams.**

Creating a Module's Call

You can link modules using calls.

You can display calls in the current diagram in the following ways:

- A: Create a call from new, i.e. a relationship didn't exist between the modules involved anywhere in the model up to now.
- B: Operations are already assigned to the modules. These operations are linked by calls in the operation diagrams. Adjust the modules using the assigned operations and, in doing so, create the appropriate calls.

C: The modules involved are already linked in another module diagram.

You want to display this relationship in the current diagram.

These calls generally make the modules involved visible using the maintenance status. For information about the maintenance status, read chapter B-2.1.3, page B-24.

If you have checked version objects into your model, inconsistencies may exist in the call structure. If this is the case, adjust the call links of the entire model and, in doing so, complete the missing call links in the current diagram.

- Read how to change the call's source or target in section "Assigning a Relationship to a New Model Element", page B-45.
- chapter B-2.5, "Removing and Deleting Elements", page B-61

Other Information

If you create a call, this call is also incorporated into all diagrams which the modules involved are displayed in, as long as no user-defined maintenance is set for these (see page B-24).

Creating New Calls

How to proceed

To create a call:

- » If necessary, lock the diagram.
The order in which you select the modules involved determines the direction of the call.
- » Select the module which should be the calling module.
- » Then select the module which should be called.
- » Select **Insert>Call**.
The modules are linked by an arrow from the calling to the called module. If one of the called modules is a connector, an arrow with an extra target connector is created instead of an arrow (see page G-67).

If inconsistencies occur regarding calls upon checking-in of version objects, you can adjust the current diagram with all of the model's other module diagrams. This completes all missing calls between your diagram's modules.

Adjusting Calls with Module Diagrams

How to proceed

To complete missing calls in the current diagram:

- » If necessary, lock the diagram.

- » Select [Engineering>Adjust>Module](#).

Calls which exist between the same modules in other module diagrams are entered.

For modules which have their maintenance status set to global, calls are used to add the linked modules in the diagram if necessary.

G-5 Preparing Implementation

Generate the necessary implementations in one of the final steps before including the results of analysis and design in the source code. You must, therefore, already have completed the following work steps:

- **Structured analysis** Specifying SA elements and data structures in SA diagrams (see page G-3), specifying control structures of real-time extension (see page G-43).
- **Structured design:** Specifying call hierarchy of functions with SD elements, setting formal interfaces and transfer parameters, merging operations in modules (see page G-57).
- **Assigning SD elements to SA elements:** This ensures that all aspects of your analysis are also implemented (see page G-66).
- **Assigning a module's operations and data fields:** This sets which source code files the individual functions are carried out in (see page G-77).

Once you have carried out all of these steps you can create the implementation. Proceed as follows:

Procedure

- Create the necessary include files (C), copy books (COBOL) and source code files.
- Assign the entries in the data dictionary as data structures to include files or copy books.
- If necessary, assign your model's module to the source code files.
- Transfer the model data to the assigned files.
- These implementations are initially stored in the repository. You can program the functions using the Innovator structogram processor or one of your own programming tools.
- You need to have exported the module from the repository to the file system to be able to compile it. Innovator management is removed for this.

If you now make changes to the source code files (in the file system), these will no longer be adjusted with the model data.

Note

However, if you continue to maintain your source codes in the Innovator repository, then your model data remains consistent and can be adjusted against each other at any time. After a repeated export of the implementation, you can recompile these.

G-5.1 Making Settings for Implementation

Before creating implementation files, you can make various specifications for transferring your design data:

- Select a target language for the design.
- Enter the file name extensions for include files, copy books or source code files.
- Set rules for the names of the source code files.
- Set how comments and program text should be generated in implementation modules and include files or copy books.
- Create shortnames for type definitions in the data dictionary.

G-5.1.1 Selecting the Implementation's Target Language

You can choose between the following target languages:

- None
- C
- COBOL

Attention If you change the target language even though appropriate implementations already exist, these implementations are lost.

The information about the previous target language's file name extension is also lost. Default settings are entered for this.

How to proceed

To enter a target language:

- » Jump to the model browser and log-in as administrator (see chapter A-1.2.1, "Logging In and Out as Model Administrator", page A-5).
- » In the model browser, select **Model>Configuration>General Settings....**

The dialog box of the same name appears.

- » Select the tab **Implementation**.
- » Select the target language you want your implementation to be created in from the drop-down list **Language**. If you use a programming language which is not in the drop-down list, select the entry **None**.
Regardless of which language you have selected, the dialog boxes which are grayed out are not required.

G-5.1.2 Influencing the Source Code Files' Names

You can set how the source code file's name and its file name extensions should be created when generating implementations from the design.

- Set the maximum length of the file name.
- Set which file name extensions the source code files to be created should have.
- Set whether case sensitivity should be taken into consideration when comparing module names.

The model data is initially converted into source code upon implementation generation, but it is not yet stored in the file system but in repository. The source code files which can be compiled are stored in the file system once these implementations have been exported.

Read how to adjust the files in the file system with the implementation in the repository at the same time in chapter G-5.3.1, "Automatically Exporting Implementations", page G-101.

How to proceed

To influence the source code file's name:

- » Jump to the model browser and log-in as administrator (see chapter A-1.2.1, "Logging In and Out as Model Administrator", page A-5)
- » In the model browser, select **Model>Configuration>General Settings...**
The dialog box of the same name appears.
- » Select the tab **Implementation**.

The names of the modules in your model are abbreviated to a length which you can set when generating the implementation. If clashes occur in the file names because of these abbreviations, you can determine where these clashes are using a verification routine and correct them if necessary.

If the only difference between the names is the case sensitivity once this abbreviation has taken place, this can also cause clashes in certain operating systems—which can ultimately lead to files being overwritten. You can make a setting so that these modules are identified as having the same name and a verification message informs you of this.

Influencing File Names

The settings do not stop you being able to use model names which will lead to name clashes during implementation generation.

Note

How to proceed

To influence the file names:

- » Set how many characters the module's name should be abbreviated to in the dialog box **Configuration/General Settings** in the group box **File restrictions** using the field **File name length**.
- » Deactivate the check box **Case sensitive** if names which are not spelled the same should cause a verification message.

Setting File Name Extensions

Information about file name extensions depends on the target language.

Attention If you change the target language, the information about the previous target language's file name extension is lost. Default settings are entered for this.

The following file name extensions are used as standard:

Target language	Include files/ copy books	Source code files
C	*.h	*.c
COBOL	*.cpy	*.cob

How to proceed

To show the default settings:

- » Select the desired target language in the dialog box **Configuration/General Settings** in the drop-down list **Language**.
- » Enter the desired file name extensions in the group box **File extensions**'s fields.
- » Close the dialog box with **[OK]**.

G-5.1.3 Controlling Generation

There are various ways to set how your model data should be transferred into the implementation. You require administrator rights for all these settings.

How to proceed

To show the default settings:

- » Jump to the model browser and log-in as administrator (see chapter A-1.2.1, "Logging In and Out as Model Administrator", page A-5).
- » In the model browser, select **Model>Configuration>General Settings....**
The dialog box of the same name appears.
- » Select the tab **Implementation**.

As far as possible, data is converted from the model into the necessary syntax and inserted into the implementation as standard. There are two ways to proceed for this:

- **Data as a comment:** Innovator makes suggestions in the implementation's comments area. You can decide whether you want to use these suggestions or not. If you want to use the suggestion, copy the comment to the right place and change it if necessary. If you make changes to the model data at a later stage, the comment is update accordingly, but the source code edited by you is not updated.
- **Data in the code:** Innovator integrates the model data into the source code. The passages are not inserted in the appropriate position as source code which can be edited. If you make changes to the model data at a later stage, the source code is update accordingly.

You can control how the following model data is generated in the source code in this way:

- **Type declarations from the data dictionary:** They are not generated as comments in the include files or copy books; they are generated as program text. The include instructions of these files also appear in the modules and are not just noted as comments. The type declarations are explicitly entered in the linkage or working storage section or using the copybook for the target language COBOL and not as a comment.
- **Formal interfaces:** Formal interfaces from the SD are generated as program text rather than comments.
- **Forward declarations:** (Only for C) forward declarations are generated as program text rather than comments.

Model Data as Comments or Source Code

How to proceed

To set whether the model data should be suggested as comments or integrated as set in the source code:

- » Select the desired target language in the dialog box **Configuration/General Settings** in the drop-down list **Language**.
- » Use the check boxes in the group box **Language** to set how you want to use the type declarations from the data dictionary and/or the formal interface.

- » If you have selected C as the target language, set how the forward declarations should be dealt with using the check box **Forwards to code** in the group box **C**.

References from DD Entries in Include Files

If you have selected C as the target language, you can set whether the comments created upon generation of type definitions for DD entries in the created comment's include files also contain type definitions to all referenced data dictionary entries from the corresponding data dictionary entry.

How to proceed

To fully transfer the data dictionary entries into the comments:

- » Activate the check box **Exploded types** in the dialog box **Configuration/General Settings** in the group box **C**.

Interrupting Different Designations of Names and Types

For C compilers who require different designations of names and types for data dictionary entries can define a postfix for the types in their model.

This postfix is automatically attached to the component names of the structures (**struct** or **union**) upon implementation generation. This means that you can avoid compiler messages even if you use the same names and types in the model.

How to proceed

To provide all type designations with a postfix:

- » Enter the character which should automatically be attached to the type definition in the dialog box **Configuration/General Settings** in the group box **C** in the field **Typedef suffix**.

Controlling COBOL Generation

You can make certain specifications for the COPY command when transferring model data to COBOL:

- **Copybook prefix:** Set a prefix which will be added in front of each data field name within a copy book.
- **COPY command instead of type declaration:** Set whether copy commands should be used in the linkage or working storage section instead of explicit type declarations.
- **Name of the copy book in inverted commas:** Set whether the name of the copy book should be generated for the COPY command in inverted commas.

How to proceed

To make settings for COBOL generation:

- » Enter the character row which should be added to the front of the data field names in the copy book in the dialog box **Configuration/General Settings** in the tab **Implementation** in the field **Copy-book prefix**.
- » Activate the check box **COPY generation** to generate a copy book.
- » Activate the check box **Copybook quotes** to generate the copy book in inverted commas.

How to proceed

Once you have made all settings:

- » Close the dialog box with **[OK]**.

G-5.1.4 Using Shortnames for DD Entries

You have specified the data structures in the data dictionary based on the structured analysis' data flows. These data structures are adopted directly in the include files or copy books upon implementation.

Designations from the analysis are often so long that you would not want to use them for the implementation or they cannot be translated. You can store a suitable shortname for each entry in the data dictionary. The detailed analysis name for the type definition is, therefore, not used when generating the implementation, but this shortname instead.

Enter the short name in the data dictionary's comment.

Syntaxes for Shortnames

Comments are contained within the entry's definition using asterisks (*). Characters within these asterisks are not included in the type definition.

Note

However, if Innovator comes across angle brackets (< >) in a comment, the text within these brackets are used as the shortname during the implementation instead of the entry itself. Text outside of the angle brackets (but within the asterisks) apply as comments.

```
D bank code = "BANKCODE" *<BC> shortname* .
```

How to proceed

In order to get shortnames in the implementation instead of the DD entries' long analysis names:

- » Open the DD entry in the data dictionary editor as described on page G-26.
- » Enter the shortname in angle brackets within the comment.
- » Save in the DD editor.

The DD entry itself is no longer used as the type definition when generating the implementation, but this shortname instead.

G-5.2 Creating Implementations

If you have made implementation settings (see page G-86), create files to transfer the model data into.

Create include files (in C) or copy books (in COBOL) for data dictionary entries and source code files for the modules.

Note **The files are initially created in the repository. You cannot compile the files here. Only when the files are exported onto the file system do you obtain source code files which can be compiled.**

Assign the desired DD entries to the include file or copy book. The files are automatically maintained from this point in time, i.e. each change made to the DD entry—whether it be the name or its definition—is automatically updated.

The source code files, on the other hand, refer to precisely one model, one operation or one data area in the model. If you change the element in the model, you need to explicitly adjust the source code file.

G-5.2.1 Creating Include Files or Copy Books

If you want to create include files or copy books for data structures, create an include file or a copy book in an address package (see page A-40). As a second procedure, you still also have to assign the DD entry/entries to this file.

Creating Include Files or Copy Books in the Model Browser

You can create an include file or a copy book directly in the model browser. After this, assign the desired DD entries to this include file or copy book.

Read how to create address packages in section "Displaying the Directory Structure from File Systems", page A-40.

How to proceed

To create an include file or a copy book in an address package:

- » Jump to the model browser and select the address package which you want to create the files in in the model tree.
- » Select **Element>New>Implementation>Include**.
An include file or a copy book with a name set automatically as standard is created. It contains the file name extension which you determined in the settings for the implementation (see page G-88). Read how to assign DD entries to the include file or copy book in chapter G-5.2.2, page G-93.

You can rename include files or copy books.

DD entries which are already assigned to files also remain assigned.

Renaming an Include File or a Copy Book

How to proceed

To rename an include file or a copy book:

- » If necessary, jump to the model browser and select the address package in the model tree where the files are located.
- » Select the include file or the copy book which you want to rename and lock it.
- » Select **Element>Properties...**
The dialog box of the same name appears. The current file name is shown in the field **Name**.
- » Enter the new name of the include file or the copy book in the field **Name** and close the dialog box with **[OK]**.
the file is renamed.

G-5.2.2 To assign a DD entry to an include file or a copy book:

When you have created an include file or a copy section, as described on page G-92, you can assign it the desired DD entries.

A DD entry cannot be assigned to more than one include file or copy book. There are three possible ways to do this:

A: In the data dictionary editor, enter the DD entry/entries, as well as the include file or copy book, which you want to assign the entries to.

B: In the model browser, assign the selected DD entries to an include file or copy book which already exist (see page G-94).

C: In the model browser, assign the designated DD entries to a selected include file or copy book.

Note **In all cases, the include files or copy books must not be locked; they must also not be locked by you.**

As soon as the DD entries have been assigned an include file or copy book, they are automatically maintained, i.e. each change to a DD entry (whether it be its name or definition) is automatically updated in the include file or copy book.

Making Assignments Via the Data Dictionary Editor

The entries must not be loaded when assigning DD entries to an include file or a copy book in the data dictionary editor.

How to proceed

To assign a DD entry to an include file or a copy book:

- » Open the data dictionary editor as described on page G-26.
- » Select **Engineering>Assign to....**
The dialog box of the same name appears. All DD entries are listed in the drop-down list **Name**.
- » You can restrict the DD entries list by **Data**, **Control** or **View** type. Using the check box in the group box **What**, enter which types of DD entries you wish you restrict the list by.
- » Select the entry from the drop-down list **Name** which you want to assign to an include file or copy book. If necessary, use a text sample to limit the selection
All include files or copy books which are already created in the repository are listed in the drop-down list **Include**.
- » Select the name of the include file in the field **Include**.
- » If you also want to assign DD entries to include files or copy books which are needed in addition to the ones already selected, activate the check box **Complete**.
- » Close the dialog box with **[OK]**.
The DD entries are assigned to the files.

Assigning Selected DD Entries to Files

You can select DD entries in the model browser and assign them to an include file or copy book. The files need to already exist in the repository when proceeding in this way.

How to proceed

To assign a DD entry to an include file or a copy book:

- » Jump to the model browser and select the package where the DD entries are stored in the model tree.
The DD entries are listed in the list of model elements.
- » Select the DD entries in the list of model elements which you want to assign to a file.
- » Select **Engineering>Assign to....**
The dialog box of the same name appears.
- » Select the tab **Implementation Assignment**.
All include files or copy books which are already created in the repository are listed in the list box **File**.
- » If necessary, restrict the list by entering a text sample in the field **Filter**.
- » Select the file from the list box **File** which you want to assign the DD entries to.
- » If you also want to assign DD entries to include files or copy books which are needed in addition to the ones already selected, activate the check box **Complete**.
- » Click on **[Assign]** and close the dialog box with **[Close]**.
The DD entries are assigned to the files.

You can assign DD entries in the model browser to an include file or copy book.

Assigning DD Entries to Selected Files

How to proceed

To assign a DD entry to an include file or a copy book:

- » Jump to the model browser and select the address package in the model tree.
- » Select the include file or copy book in the list of model elements which you want to assign DD entries to.
The file must not be locked—also not by you.
- » If necessary, unlock the file.
- » Select **Engineering>Assign to....**
The dialog box of the same name appears.
- » Select the tab **DD Assignment**.
All of the DD entries which are not yet assigned to the include file or copy book are listed in the list box **DD entries**. You can restrict the DD entries list by **Data**, **Control** or **View** type.
- » Using the check box in the group box **Type**, enter which types of DD entries you wish to restrict the list by.

- » If necessary, further restrict the list by entering a text sample in the field **Filter**.
- » Select the DD entries in the list which you want to assign to the file.
- » If you also want to assign DD entries to include files or copy books which are needed in addition to the ones already selected, activate the check box **Complete**.
- » Click on **[Assign]** and close the dialog box with **[Close]**.
The DD entries are assigned to the files.

Removing Assignments

You can remove the assignment of a DD entry to an include file or copy book. This also deletes the entry's type definition from the file.

How to proceed

To remove an assignment from an include file or a copy book:

- » Jump to the model browser and select the package where the DD entries are stored in the model tree.
The DD entries are listed in the list of model elements.
- » Select the DD entries in the list of model elements which have an assignment to a file you wish to remove.
- » Select **Engineering>Assign to...**
The dialog box of the same name appears.
- » Select the tab **Implementation Assignment**.
All include files or copy books which are already created in the repository are listed in the list box **File**. If all selected DD entries are assigned to the same file, the corresponding entry is selected in the list.
- » Click on **[Remove]** and close the dialog box with **[Close]**.
The assignment of the DD entries to the files is removed.

G-5.2.3 Creating Source Code Files and Transferring Model Data to the Implementation

Proceed as follows when creating header and source code files:

- Create empty files (see page G-92) when creating **include files or copy books** and then the desired DD entries (see page G-93).
- Create **source code files** by adjusting the implementation for a module, an operation or a data area. This creates a source code file for each (adjusted) element. The files contain the name of the element (if applicable, abbreviated) in the predefined length (see page G-87).

In this chapter you will learn how to create source code files and adjust them with the model data.

Only the files are updated in the repository when the source code files are created and the implementation is adjusted. You then need to export the implementation into the file system to be able to compile changes.

Note

Read how to save the files in the file system with the implementation in the repository at the same time in chapter G-5.3.1, "Automatically Exporting Implementations", page G-101.

If you have created source code files using the adjustment, a fixed relationship between the files and the respective model elements exists. Each subsequent adjustment automatically updates the relevant files in the repository.

- chapter G-5.1.2, "Influencing the Source Code Files' Names", page G-87
- section "Displaying the Directory Structure from File Systems", page A-40
- chapter G-5.2.4, "Creating Other Source Code Files", page G-98

Other Information

You can create source code files for internal modules, internal operations and internal data fields.

Creating the Source Code Files

If your target language is C, you can merge your model's operations in modules (see page G-77).

- Each of these modules contains a source code file.
- If you want to create a source code file for an operation which is already assigned to a module, create a source code file for the module and not for the operation.
- If the operation is not assigned to a module (e.g. in the target language COBOL), then create a source code file for the operation.

Source code files can be created both via the model browser and one of the diagrams which is showing the respective model element.

How to proceed

To create source code files in the model browser:

- » Jump to the model browser and select the package from the model tree which contains the model element you wish to create a source code file for.
- » Select the element/elements you wish to create the source code files for and select [Engineering>Implementation>Adjust](#).

The source code files are created for the selected elements in the address package. The files contain the names of the elements (if applicable, abbreviated) in the predefined length (see page G-87). The model data is transferred to the files.

How to proceed

To create source code files in an operation or module diagram:

- » Jump to the diagram which will display the element.
- » Select the element/elements you wish to create the source code files for and select [Engineering>Adjust>Implementation](#).

The source code files are created for the selected elements in the address package. The files contain the names of the elements (if applicable, abbreviated) in the predefined length (see page G-87). The model data is transferred to the files.

Updating Existing Source Code Files

If you make changes to model elements which already have implementations (e.g. assigning other operations to a module), you need to adjust the implementation. This action also updates the linked source code files in the repository.

Note If you set that interface definitions and type definitions are only written in the implementation's comments for your model (see page G-89), only the comment is updated upon adjustment. You then need to manually reproduce the changes in the code too.

Source code files can be adjusted both in the model browser and in one of the diagrams which displays the respective model element. The procedure is exactly the same as when creating a source code file (see page G-97).

Note If you want to adjust a source code file, this cannot be locked—it can also not be locked by you.

G-5.2.4 Creating Other Source Code Files

If you need implementation for programs which you do not maintain in your model's structured design, you can create implementation files in addition to these.

Note You cannot assign modules, operations or data areas from the model to these files.

You can create a file for modules, operations and data.

Creating Source Code Files

You can create source code files directly in an address package (see page A-40) in the model browser.

How to proceed

To create a source code file in an address package:

- » Jump to the model browser and select the address package in the model tree.
- » Select **Element>New>Implementation><Element Type>**.
A source code file with a default name is created. It contains the file name extension which you determined in the settings for the implementation (see page G-88).

You can rename the definition files. Model elements which are already assigned to the files still remain assigned to them.

Renaming Source Code Files

How to proceed

To rename a source code file:

- » Jump to the model browser and select the address package in the model tree.
- » Select the source code file which you want to rename and lock it as well.
- » Select **Element>Properties....**
The dialog box of the same name appears. The current source code file's name is shown in the field **Name**.
- » Enter the new name of the source code file in the field **Name** and close the dialog box with **[OK]**.
The source code file is renamed.

G-5.2.5 Opening Implementations

You can open and edit the header and source code files which you created for the model data in the repository.

You open the implementations using the structogram processor in Innovator as standard. However, if you have integrated another editor, this is automatically opened with the appropriate implementation.

You can edit and save any number of data types in the implementation for the current model in the programming language used.

There are two ways to open an implementation:

A: If you know which file an element's implementation is in, select the file directly in the model browser and open it.

B: Select an element in a diagram and jump to its implementation.

If e.g. you want to open an operation's implementation which is already assigned to a module, select the operation in the OP diagram and jump to the module's implementation.

Opening implementations from the Model Browser

All implementations which already exist in the repository can be found in the address package in the model browser.

How to proceed

To open an implementation in the model browser:

- » Open the address package in the model browser.
All files which already exist for the model data are listed in the list of model elements.
- » Select the file which you want to open and select [Element>Open](#).
The editor set for your system is opened with the selected implementation.

Opening Implementations from the Diagram

Open an operation's implementation from the operation diagram. If the operation is not yet assigned to a module, in doing so you open the file which only contains the operation. If the operation is already assigned to a module on the other hand open the module's source code file.

Open a module's implementation from the module diagram.

How to proceed

To open an operation's implementation or a module's implementation:

- » Jump to the diagram which contains the model element.
- » Select the model element. Ensure that you are not trying to select a pre-defined operation or module.
- » Select [Jump>Reference...](#)
The dialog box of the same name appears. If you have already adjusted the model element with the implementation (see page G-96), the [Operation Implementation](#) or [Module Implementation](#) entry appears in the column [Element type](#).
- » Select the entry.
The corresponding implementation is already selected in the column [Name](#).
- » Close the dialog box with [\[Jump\]](#).
The editor set for your system is opened with the selected implementation.

Opening Include Files or Copy Books for Data Types of SD Operations

The include files or copy books are not assigned to any module or operation. You can open any number of include files or copy books from the diagrams.

How to proceed

To open an include file or copy book from an operation or module diagram:

- » Remove all selections from the diagram and select **Jump>Reference...**.
The dialog box of the same name appears. If include files or copy books already exist in the repository (see page G-92), the **Include Implementation** entry can be seen in the column **Element type**.
- » Select the entry.
All of the model's include files or copy books are listed in the column **Name**.
- » Select the file which you want to open in the column **Name** and close the dialog box with **[Jump]**.
The editor set for your system is opened with the selected file.

G-5.3 Exporting Implementations in the File System

If you have transferred the model data into the implementation as described in chapter G-5.2, page G-92, you can export these in the file system. Once this is done, the implementation is in a compatible format.

You can export the implementation in a targeted manner for certain implementations or automate this process. This means that when changes made to an implementation are saved in the Innovator editor (Nassi-Shneiderman editor), they are saved both in the repository and on the file system.

The directory which the source code files should be created in must be known for the implementation's export. Read how to specify address packages in section "Displaying the Directory Structure from File Systems", page A-40.

G-5.3.1 Automatically Exporting Implementations

You can make settings so that the corresponding files are also updated in the file system when you save the implementation in the Innovator editor (Nassi-Shneiderman editor).

How to proceed

To export the implementations in the file system at the same time as saving them:

- » Jump to the model browser and log-in as administrator (see chapter A-1.2.1, "Logging In and Out as Model Administrator", page A-5)
- » In the model browser, select **Model>Configuration>General Settings...**

The dialog box of the same name appears.

- » Select the tab **Implementation**.
- » Activate the check box **Lifelink** in the group box **File restrictions**.

The **Lifelink** option controls whether changes made to an implementation or an external object are saved both in the repository and on the file system when saved in the Innovator editor (Nassi-Shneiderman editor (**inosp.exe**) or external object manager application (**inoexo.exe**)).

- » Close the dialog box with **[OK]**.

G-5.3.2 Manually Exporting Implementations

If automatic export of implementations is not activated for your model (see page G-101) you need to manually export the adjusted implementation from the repository into the file system to be able to compile them there.

How to proceed

To manually export the implementation:

- » Jump to the model browser and select the address package in the model tree.
- » Select the entries which you want to export in the list of model elements.
- » Ensure that the desired files are adjusted with the most up-to-date model data (see page G-96).
- » Ensure that a working directory is defined for the address package (see page A-40).
- » Select **Engineering>Implementation>Export**.

The implementations are exported from the repository into the directory which is set as the working directory in the address package.

Index

A

- Action **G-49**
 - Entering G-49
- Activating
 - Process G-54
- Adding
 - Data area G-60
 - Element G-7, G-60, G-81
 - Module G-81
 - Operation G-60
- Adjusting
 - Call G-82
- Adopting
 - Formal parameters as the current parameters G-75
- Arguments
 - Enabling variable ~ G-66
- Assigning
 - Assigning Include Files to a Data Dictionary Entry G-95
 - Data dictionary entry
 - An include file G-93
 - SD elements to a module G-77
 - SD elements to SA elements G-66
- Assignment
 - Removing G-96
 - Removing o SD module G-78

B

- Bi-directional
 - Flow G-12
 - ~ Creating G-22
 - ~ Splitting G-23

C

- Call **G-71, G-82, G-82**
 - Adjusting
 - With module diagrams G-83
 - Appending missing ~ G-73
 - Creating G-72, G-82
 - Creating as new G-83
 - Creating for start connectors G-70
 - see Call
 - Setting the order G-73
- Changing
 - Current parameter G-76
 - Flow G-15
 - Flow type G-18
 - Formal parameter G-65
 - Transition G-48
- Changing the order
 - Call G-73
 - Current parameter G-76
 - Formal parameter G-65
- Changing the source
 - Flow G-24
- Changing the target
 - Flow G-24
- COBOL
 - Controlling generation G-90
- Connector **G-67**
 - Creating
 - For a call G-69
 - For operations G-69
 - Following G-71
 - Renaming G-70
 - Start~ G-68
 - Switching from start~ to target connectors G-71
 - Switching from target~ to start connectors G-71
 - Target~ G-68
- Context diagram **G-3**
- Control flow **G-11**
 - Changing to data flow G-18
- Control parameter G-74
- Copying
 - Data area G-60
 - Element G-60, G-81
 - Module G-81
 - Operation G-60
- Creating
 - Bi-directional flow G-22
 - Call G-72, G-82
 - Connector
 - For a call G-69
 - For operations G-69
 - Definition file G-97, G-98
 - In the diagram G-98
 - In the model browser G-97
 - Element G-82
 - Flow G-13
 - Include file
 - In the model browser G-92
 - Input rows (PAT) G-53
 - Merge flow G-21
 - Module diagram G-79
 - Operation diagram G-58
 - Output columns (PAT) G-53
 - Parameter G-63, G-74
 - Process Activation Table G-50
 - Process columns (PAT) G-52
 - RT decision table
 - Using SA diagrams G-44
 - SA Decision Table G-35

- Split flow G-20
- State G-47
- State diagram G-46
- Time requirements table G-56
- Transition G-48
- Current parameter G-74
 - Changing G-76
 - Changing the order G-76
 - Creating G-74

D

- Data area **G-58**
 - Adding G-60
 - Copying G-60
 - Defining interfaces G-63
 - External G-59
 - Internal G-59
 - Opening implementations G-99
 - Parameter
 - Adopting G-63
 - Changing G-65
 - Changing the order G-65
 - Creating G-63
 - Defining G-63
 - Deleting G-65
 - Transforming externals ~ into internals ~ G-62
- Data dictionary **G-25**
 - Entry
 - Assigning an include file G-93
 - Transferring in the comments G-88, G-90
 - Using shortnames G-91
 - Entry as flow name G-17
 - Literal G-29
 - Syntax G-28
- Data dictionary editor **G-27**
 - Emptying the window G-27
 - Entry
 - Complete G-27
 - Correcting G-33
 - Deleting G-34
 - Detailed information G-31
 - Finding undefined ~ G-32
 - Inserting G-28
 - Opening Summary G-30
 - Renaming G-33
 - Saving G-32
 - simple G-27
 - Verify G-32
 - Identifying the modification status G-30

- Importing views from (S)ERM G-25
- Opening G-26
- Data Flow **G-11**
- Data flow
 - Changing to control flow G-18
- Data parameter G-74
- Dataflow diagram
 - See SA diagram
- DD
 - See data dictionary
- DDE
 - See data dictionary editor
- Decomposition **G-3**
 - In the data dictionary G-17
- Defining interfaces
 - Of a data area G-63
 - Of an operation G-63
- Definition file
 - Creating G-97, G-98
 - In the diagram G-98
 - In the model browser G-97
 - Updating G-98
- Deleting
 - Entries in the data dictionary G-34
 - Formal parameter G-65
- Direction
 - Changing
 - Flow G-13
 - Transition G-48

- E**
- Element **G-5**
 - Adding G-7, G-60, G-81
 - Assigning
 - To an SD Module G-77
 - Copying G-60, G-81
 - Creating G-82
 - Limiting the Number G-5
 - Specifying G-34
- Element types
 - Data area G-58
 - Module diagram G-79
 - Operation G-58
 - Process G-5
 - RT Decision Table G-43
 - SA Decision Table G-35
 - SA diagram G-3
 - State G-46, G-47
 - State diagram G-45
 - Terminator G-5
 - Time requirements table G-55
- Enabling variable arguments G-66
- Entering output G-55
- ERM/SERM
 - Linking with SA G-25
- Event **G-49**, G-49

- Entering G-49
- Exporting
 - Implementation
 - Automatically G-101
 - Manually G-102

F

- File names
 - Influencing G-87
 - Setting extensions G-88
- Finding
 - Undefined entries in the data dictionary G-32
- Flow
 - Bi-directional ~ G-12
 - Changing G-15
 - Changing the direction G-13
 - Changing the type G-18
 - Creating G-13
 - Data Dictionary Entries as Names G-17
 - Display G-12
 - Environment flow G-11
 - Not enabled ~ G-13
 - Reconnecting G-24
 - Refining G-11
 - Renaming G-15

H

- Header file
 - See Include file
- Hiding
 - Column groups G-52
- Hiding Column Groups G-52
- History G-30, G-31
- Hybrid parameter G-74

I

- Implementation
 - Creating G-92
 - Default settings G-88
 - Exporting
 - Automatically G-101
 - Manually G-102
 - Making settings G-86
 - Opening G-99
 - In the diagram G-100
 - In the model browser G-100
 - Preparing G-85
 - Selecting target language G-86
 - Transferring model data G-89
- Include file
 - Assigning to a data dictionary G-95

- Creating
 - In the model browser G-92
- Opening
 - For SD operation G-100
- Renaming G-93

Input rows

- Creating (PAT) G-53

Inserting

- Entries in the data dictionary G-28

L

Literal G-29

M

- Merge flow
 - Creating G-21
 - Splitting G-23
- MOD
 - See Module diagram
- Modifying
 - State G-47
- Module **G-80**
 - Adding G-81
 - Assigning SD elements to a ~ G-77
 - Copying G-81
 - Opening implementations G-99
 - Transforming external into internal G-81
- Module diagram **G-79**
 - Creating G-79
 - Elements
 - Call G-82
 - Module G-80

O

- OPD
 - See Operation diagram
- Opening
 - Data dictionary editor G-26
 - Implementation G-99
 - Include file
 - For SD operation G-100
 - Process Activation Table
 - Via the diagram G-51
 - Via the model browser G-51
 - RT decision table
 - Via the diagram G-44
 - Via the model browser G-45
- Operation **G-58**, G-63
 - Adding G-60
 - Copying G-60
 - Creating calls G-72
 - Defining interfaces G-63

- External G-58
- Internal G-58
- Main G-61
- Opening implementations G-99
- Parameter
 - Changing G-65
 - Changing the order G-65
 - Creating G-63
 - Defining G-63
 - Deleting G-65
- Transforming external ~ into internal ~ G-62
- Operation Diagram **G-57**
- Operation diagram
 - Creating G-58
 - Elements
 - Call
 - Current parameter G-74
 - Data area G-58
 - Operation G-58
- Output
 - Entering G-55
- Output columns
 - Creating (PAT) G-53

P

- Parameter
 - Adopting
 - Current as formal parameter G-63
 - Formal parameters as the current parameters G-75
 - Changing G-65
 - Changing the order G-65
 - Control~ G-74
 - Creating G-63, G-74
 - Current~ G-74, G-76
 - Data~ G-74
 - Deleting G-65
 - Hybrid~ G-74
- Process **G-5**
 - Activating G-54
 - Adding G-7
 - Changing
 - Numbering G-8
 - Order G-9
 - Creating columns (PAT) G-52
 - Limiting the number G-5
 - Specifying G-34
- Process Activation Table
 - Creating G-50
 - Opening
 - Via the diagram G-51
 - Via the model browser G-51
- Process activation table
 - Creating output columns G-53

Q

- Qualifier
 - Assigning a flow G-16

R

- Real-time Extension **G-43**
- Reconnecting
 - Flow G-24
- Refinement **G-3**
 - In the data dictionary G-17
- Refining
 - Flow G-11
- Removing
 - Assignment G-96
- Renaming
 - Connector G-70
 - Definition file G-99
 - Entries in the data dictionary G-33
 - Flow G-15
 - Include file G-93

RT

- See real-time extension

RT Decision Table **G-43**

- RT decision table
 - Creating
 - Using SA diagrams G-44
 - Opening
 - Via the diagram G-44
 - Via the model browser G-45

S

- SA
 - See structured analysis
- SA Decision Table **G-35**
 - Creating G-35
- SA diagram **G-3**
 - Changing max. no. of processes G-6
 - Elements G-3
 - Control flow G-4
 - Data Flow G-4
 - Process G-5
 - Terminator G-5

Saving

- Entries in the data dictionary G-32

SD

- See Structured design

SD element

- Assigning
 - To a module G-77
 - To an SA element G-66

- Selecting the target language for implementation G-86

- Shortnames in data dictionary G-91

Source code file

- Influencing names G-87

Specifying

- Process G-34

Split flow

- Creating G-20
- Splitting G-23

Start connector **G-68**

- Switching to target connectors G-71

State **G-46**

- Creating G-47
- Modifying G-47

State diagram **G-45**

- Creating G-46

State transition **G-47**

Store

- Adding G-7

Structured analysis **G-43**

- Application field G-3

Structured design G-57

Syntax

- In the data dictionary G-28

T

Target connector **G-68**

- Switching for start connectors G-71

Terminator **G-5**

- Adding G-7
- Permitting in flow diagrams G-6

Time requirements table **G-55**

- Creating G-56

- Transferring model data in the implementation G-89

Transition **G-47**

- Changing G-48
- Changing the direction G-48
- Creating G-48
- Entering
 - Action G-49
 - Event G-49

TRT

- see Time requirements table

U

Updating

- Definition file G-98

V

Verify

- Entries in the data dictionary G-32

MID GmbH

Headquarters Nuremberg
Eibacher Hauptstrasse 141
90451 Nuremberg
Germany

Tel.: +49 (0)911 96836-22
Fax: +49 (0)911 96836-10

E-Mail: info@mid.de
www.mid.de

Branch Cologne
Ettore-Bugatti-Strasse 6-14
51149 Cologne
Germany

Tel.: +49 (0)2203 20271-0
Fax: +49 (0)2203 20271-10

Branch Stuttgart
Silberburgstrasse 187
70178 Stuttgart
Germany

Tel.: +49 (0)711 633859-0
Fax: +49 (0)711 633859-10

Branch Munich
Keltenring 7
82041 Oberhaching
Germany

Tel.: +49 (0)89 95476831-0
Fax: +49 (0)89 95476831-9